encapsulation based on multi-message. A M-KEM scheme consists of the following five algorithms:

1). Setup: k is a security parameter, the algorithm inputs 1^k, and outputs system parameters *params*.

2). The sender key generation algorithm (*SKeyGen*): the algorithm inputs *params*, and outputs the sender's public / private key pair (pk_s, sk_s).

3). The receivers key generation algorithm (*RKeyGen*): the algorithm inputs *params*, and outputs the receivers' public / private key pairs $(pk_{r1}, sk_{r1}), (pk_{r2}, sk_{r2}), ..., (pk_{rn}, sk_{rn})$.

4). Key encapsulation algorithm (*Encap*): this algorithm inputs system parameters *params*, the private key sk_s, message m and (pk_{r1}, pk_{r2},......, pk_{rn}) , and outputs symmetric key K and key encapsulation ω, where m is obtained by blending with many different messages.

5). Key decapsulation algorithm (Decap): This algorithm inputs system parameters params, a sender's public key pk$_s$, receiver's private key sk$_{ri}$ $(1 \le i \le n)$ and a key encapsulation ω, and outputs a symmetric key K or an error symbol " $\perp$ ".

4.2 Data Encapsulation Mechanism (DEM)

The formal analysis of the KEM-DEM originates from Gramer and Shoup's work [5], the key to the KEM-DEM lies in separating the cryptosystem from the different components, which allows modular design cryptosystem. Since then, many expanded or improved KEM mechanism have been proposed [6-8], and the DEM is still a symmetric encryption technique and keeps the original DEM definition. Therefore, this paper also uses the definition of DEM in reference [5], and it is described as follows:

1). Symmetric encryption algorithm (Enc): this algorithm inputs a symmetric key K and the message m, where m is obtained by blending with many different messages $m_1, ..., m_n$, and outputs a ciphertext $C = Enc_K(m)$.

2). Symmetric decryption algorithm (Dec): this algorithm inputs a symmetric key K and the ciphtext C, and outputs a message $m = Dec_K(C)$.

4.3 Definitions of Multi-message and Multi-Receiver Hybrid Signcryption (M-HSC)

The main purpose of the new scheme is confidentially and authentically to broadcast several different messages to multiple receivers, and ensures each receiver is fair, alone unsigncrypt message to get their own plaintext. The M-HSC scheme consists of M-KEM, DEM, setup, *SKeyGen*, *RKeyGen*, signcrypt and unsigncrypt. The detail is described as follows:

Setup: same as the *setup* algorithm in the M-KEM scheme;

SKeyGen: same as the *SKeyGen* algorithm in the M-KEM scheme;

RKeyGen: same as the *RKeyGen* algorithm in the M-KEM scheme;

Signcrypt: inputs $(params, pk_s, sk_s, pk_{r1}, pk_{r2}, \cdots\cdots, pk_{rn})$ and messages $m_1, \cdots, m_n$, where m_i will be send to the i receiver for $1 \le i \le n$. The sender calculates ciphertext σ by performing the following steps.

1). Blend n message $m_1, \cdots, m_2$ to get message m, where m$_i$ is corresponding to the receiver i.

2). Compute (K, ω) by using the *Encap* algorithm of M-KEM;

3). Compute ciphertext C by using the *Enc* algorithm of DEM;

4). Output $\sigma \leftarrow (C, \omega)$.

Unsigncrypt: Inputs $(params, \sigma, pk_s, pk_{ri}, sk_{ri})(1 \le \mathrm{i} \le n)$, and each receiver performs the following steps.

1). Compute K by using the *Decap* algorithm of M-KEM;

2). Compute m by using the *Dec* algorithm of DEM;

3). Check whether the relevant verification equation is hold or not. If it is true, the receiver accepts *m*, otherwise outputs symbol " $\perp$ ";

4). Using its private key, the receiver calculates x$_i$, and then obtains message m$_i$.

5. A New Multi-message and Multi-receiver Hybrid Signcryption Scheme

5.1 The proposed scheme

In this section, we present an efficient and secure multi-message and multi-receiver hybrid signcryption scheme based on the discrete logarithm. The following shows the details of the scheme.

Setup: Input a security parameter $k \in$ N, and KGC chooses the system parameters which include cyclic group G_1 of prime order $q \ge 2^k$, a generator $g \in G_1$ and a large prime number p. The KGC also chooses cryptographic hash functions $h_1 : \{0,1\}^l \to G_1$, $h_2 : \{0,1\}^{l^*} \to G_1$, $h_3 : \{0,1\}^l \to G_1$, where l is the length of the computed hash, and l^* is the length of the key. The system parameters are $params = (G_1, g, q, p, h_1, h_2, h_3)$.

SKeyGen: Given *params*, KGC generates a public/ private key pair of sender (pk_s, sk_s), where $sk_s \in Z_P^*$, $pk_s = g^{sk_s} \bmod p \in Z_P^*$, and the (pk_s, sk_s) is sent to the sender and pk_s is public.

RKeyGen: Given *params*, KGC generates *n* public / private key pairs of receivers (pk_{r1}, sk_{r1}), $(pk_{r2}, sk_{r2}), \ldots\ldots, (pk_{rn}, sk_{rn})$, where $sk_{ri} \in Z_P^*$, $pk_{ri} = g^{sk_{ri}} \in Z_P^*$, （$i=1,2,\ldots\ldots,n$）, and KGC send these *n* public / private key pairs to the appropriate receivers and publish all public key $pk_{r1}, pk_{r2}, \ldots\ldots, pk_{rn}$.

Signcrypt: Suppose a sender wants to signcrypt messages $m_1, m_2, \ldots\ldots, m_n$ to *n* different receivers, and the sender does the followings:

1). Choose r, $x \in Z_P^*$ and compute $\delta = g^r \bmod p$.

2). Compute $x_i = h_1(pk_{ri}^r \cdot sk_s \bmod p)$, $i=1,2,\ldots,n$.

3). Compute the message to be sent, $M = (m_1 \oplus x_1) \| \cdots\cdots \| (m_n \oplus x_n)$, where m_i is the message which will be sent to receiver *i*.

4). Compute $y = g^x \bmod p$ and $K = h_2(y)$.

5). Compute $C = Enc_K(M)$.

6). Compute $v = h_3(M) \oplus y \bmod p$.

7). Compute $s = x/(v + sk_s) \bmod p$.

8). Compute $t = y - sk_s \cdot v \bmod p$.

9). Output ciphertext $\sigma = (C, v, s, t, \delta)$.

Unsigncrypt: When receiving ciphertext $\sigma = (C, v, s, t, \delta)$, the receiver *i* follows the steps below.

1). Compute $y = (pk_s \cdot g^v)^s \bmod p$.

2). Compute $K = h_2(y)$, $M = Dec_K(C)$.

3). Compute $v' = h_3(M) \oplus y \bmod p$ to decide whether $v' = v$ holds or not. If they are equal, the receiver *i* does the following steps; otherwise, the communication is stopped.

4). Compute $x_i = h_1(\delta^{sk_{ri}} \cdot \frac{y-t}{v} \bmod p)$.

5). Find the corresponding section of message, and decrypt the message belonging to receiver *i*, $m_i = (m_i \oplus x_i) \oplus x_i$.

5.2 Correctness

It is obvious that the above unsigncryption algorithm is valid. The unsigncryption of our scheme is corrected by the following :

For each *i* with $1 \le i \le n$, we have the $y = (pk_s \cdot g^v)^s \bmod p$ and $K = h_2(y)$, thus, $M = Dec_K(C)$, and $v' = h_3(M) \oplus y \bmod p$, if $v = v'$ then the message m is correct.

Furthermore:

$$x_i = h_1(\delta^{sk_{ri}} \cdot \frac{y-t}{v} \bmod p)$$

$$= h_1(\delta^{sk_{ri}} \cdot sk_s \bmod p)$$

$$= h_1(pk_{ri}^r \cdot sk_s \bmod p)$$

Get $m_i = (m_i \oplus x_i) \oplus x_i$.

6. Security Proofs

Now we prove that our scheme is IND-M-HSC-CCA2 secure, and EUF-M- HSC -CMA is secure under the GDH assumption and the DL assumption.

6.1 Confidentiality

The confidentiality is the necessary security requirement for a signcryption scheme. It means that no useful information about a plaintext message can be gleaned from the corresponding ciphtext.

Theorem 1. In the random oracle, if an adversary *A* has non-negligible advantage against the IND-M-HSC-CCA2 security of our scheme when running in time *t* and performing q_s signcryption queries, q_d unsigncryption queries and q_i(i =1,2,3) hash queries, then there is an algorithm *B* which solves the GDH problem with probability $\varepsilon' \ge \varepsilon \cdot \left\lfloor 1 - \frac{q_s(q_s + 2q_d + 2q_3 - 1) + 2q_d}{2q} \right\rfloor$ within running time $t' \le t + (q_d + q_s)O(t_1)$ where t_1 denotes the time required for one discrete logarithm operation.

Proof. We show how to build an algorithm *B* to solve the GDH problem by running the adversary *A* as a subroutine. On inputting (g, g^a, g^b), the goal of *B* is to compute g^{ab}. After the game starts, *B* randomly selects v^*, $s^* \in Z_P^*$, setting $y = pk_r^x \bmod p$, $pk_s = (g^a \cdot g^{-v^* s^*})^{\frac{1}{s^*}} \bmod p$, $pk_r = g^b \bmod p$, and the goal is to compute :

$$y^* = (pk_s \cdot g^{v^*})^{s^* sk_r} \bmod p$$

$$= ((g^a g^{-v^* s^*})^{\frac{1}{s^*}} \cdot g^{v^*})^{s^* b} \bmod p$$

$$= (g^{\frac{a}{s^*}})^{s^* b} \bmod p = g^{ab} \bmod p.$$

B can simulate the challenger to execute each phase of the IND-M-HSC-CCA2 game for A as follows:

Phase1:

At the beginning of the game, *B* sets $params = (G_1, g, q, p, h_1, h_2, h_3)$ and generates *n* public / private key pairs $(pk_{r1}, sk_{r1}), (pk_{r2}, sk_{r2}), ...,$ (pk_{rn}, sk_{rn}) , $sk_{ri} \in Z_p^*$, $pk_{ri} = g^{sk_{ri}} \in Z_p^*$, (*i=1,2,……,n*) , which are sent to *A* ,where $H_1, H_{2,} H_3$ are random oracles controlled by B.

Let L_{H_1} , $L_{H_2}^1$, $L_{H_2}^2$, $L_{H_3}^1$, $L_{H_3}^2$ be used for storing the results of the querying $h_1, h_{2,} h_3$,respectively. During the simulation, B employs a DDH oracle, and the oracle inputs three groups of elements; if they are Diffie-Hellman tuples, then outputs symbols "$\top$" , otherwise outputs " $\perp$ ". Where DDH (g^a,g^b,g^c) denotes a DDH oracle.

H$_1$ queries: *A* inputs a public key pk_{ri} $(1 \le i \le n)$ to H$_1$, and *B* checks if there exists (x_i,n_i) in L_{H_1} . If such a tuple is found, *B* answers x_i; otherwise *B* randomly selects $\eta \in Z^*$,compute $x_i = h_1(pk_{ri} \cdot \eta \bmod p)$ and put($x_{i,}i$) in L_{H_1} , and return x_i as the answer.

H$_2$ queries: For a H$_2$(y) query, *B* performs the following steps.

- If DDH(g^a, g^b, *y*) = "$\top$", then returns *y* as the answer of GDH problem;
- Otherwise, if exists (*y*, *K*) in $L_{H_2}^1$, returns *K*;
- *Otherwise, if exists (*φ *, K) in* $L_{H_2}^2$ *and DDH (*g^b*,* φ *, y)=* $\top$ *, returns K;*
- Otherwise, *B* randomly selects $K \in \kappa$, puts (*y*, *K*) into $L_{H_2}^1$ and returns *K*.

H$_3$ queries: For a H$_3$ query, *B* performs the following steps.

- If DDH (g^a, g^b , *y*) = "$\top$", then returns *y* as the answer of GDH problem;
- Otherwise, if exists (*v* ,*y*) in $L_{H_3}^1$, returns *v*;
- Otherwise, if exists (φ ,*v*, *M*) in $L_{H_3}^2$ and DDH(g^b, φ , *y*)= $\top$, returns *v* ;
- Otherwise, *B* randomly selects $v \in Z_p^*$, puts (*v*, *y*) into $L_{H_3}^1$ and returns *v*.

Signcryption queries: For a signcryption query on plaintext $(m_1, m_2, ..., m_n) \in G_1$ chosen by the adversary *A*, *B* first randomly chooses *r*, *x* ∈ Z_P^*, computes $\delta = g^r \bmod p$, runs the H$_1$ simulation process to obtain x_i $(i = 1, 2, ..., n)$ and computes $M = (m_1 \oplus x_1 \| ...... \| m_n \oplus x_n)$. Computes $y = g^x \bmod \ p$ and obtains *K* from $L_{H_2}^1$ or $L_{H_2}^2$. Then computes $C = Enc_K(M)$. Checks if there exists (*v*, *y*) in $L_{H_3}^1$ or (φ ,*v*, *M*) in $L_{H_3}^2$; if tuple is not found, the game ends; otherwise, reads *v*, computes $s = x/(v + sk_s) \bmod p$, $t = y - sk_s \cdot v \bmod p$, and returns $\sigma^* = (C, v, s, t, \delta)$ to *A*.

Unsigncryption queries: For an unsigncryption query on a ciphertext (C, v, s, t, δ) and a sender's public key pk_s, both chosen by *A*, *B* does the following:

B computes $\varphi = (pk_s \cdot g^v)^s \bmod p$.

- If $\varphi = g^a$, the $\perp$ symbol is returned to *A* and the game is stopped ;
- If $L_{H_3}^1$ contains a tuple (*y*, v') and $M = M'$, $DDH(g^b, \varphi, y)$ =$\top$, but $v \ne v'$, the $\perp$ symbol is returned to *A* and the game is stopped;
- Otherwise, if $L_{H_3}^2$ contains a tuple (φ', M, v') and $\varphi = \varphi'$, $M = M'$, but $v \ne v'$, the $\perp$ symbol is returned to *A* and the game is stopped;
- Otherwise, randomly chooses $r' \in Z_P^*$, puts (φ, M, v') into L_{H3}^2.
- If L_{H2}^1 contains a tuple (*y*,*K*), and $DDH(g^b, \varphi, y) =$ $\top$, returns *K* to *A*;
- Otherwise, if L_{H2}^2 contains a tuple (φ', *K*), and, $\varphi = \varphi'$, returns *K* to *A*;
- Otherwise, randomly chooses $K \in \kappa$, puts (φ, K) into L_{H2}^2 and returns *K*.
- Computes $M = Dec_K(C)$, obtains x_i from L_{H1},then, gets m_i.

Challenge: *A* decides to stage 1 when stop and into the challenge. *A* chooses a target plaintext ($m_0^* = \{m_1, m_2, ..., m_n\} \in G_1$, $m_1^* = \{m_1', m_2', ..., m_n'\} \in G_1$), *B* does the following:

B randomly selects $b \in \{0,1\}$ to calculate M^* , $y = pk_r^x \bmod p$. Finally, *B* generates the ciphertext $\sigma^* \leftarrow Signcrypt(m_b^*, sk_s, pk_{r1}^*, pk_{r2}^*, ..., pk_{rn}^*)$ and sends σ^* to *A*.

Phase 2: *A* makes some new queries as in the first stage with the restriction that it can't query the unsigncryption oracle with σ^* ·

Guess: *A* outputs a bit *b'* and wins if *b'*=*b*.

If *A* wins the game, then *B* can compute:

$$y = (pk_s \cdot g^v \ )^{s \cdot sk_r} \bmod p$$

$$= ((g^a g^{-v \cdot s})^{\frac{1}{s}} \cdot g^v)^{s \cdot b} \bmod p = g^{ab} \bmod p \cdot$$

The GDH problem is solved, which is inconsistent with the assumptions.

In signcryption inquiry, it may cause a conflict that *B* adds (φ, M, v) to L_{H3}^2. In addition, L_{H3}^1 and L_{H3}^2 lists have at most $q_3 + q_d$ items in the first phase. *B* will add an entry to the L_{H3}^2 in every query. For the q_s signcryption query, B will fail with a

$\frac{q_s(q_s+2q_d+2q_3-1)}{2q}$ probability. For the q_d unsigncryption inquiry, *B* may reject a legitimate ciphertext, the probability of occurrence of this event is at most q_d/q.Therefore, the total probability which *B* failure is $\frac{q_s(q_s+2q_d+2q_3-1)+2q_d}{2q}$.Consequently, B solves the GDH problem with probability $\varepsilon' \ge \varepsilon \cdot \left\lfloor 1-\frac{q_s(q_s+2q_d+2q_3-1)+2q_d}{2q} \right\rfloor$.

6.2 Unforgeability

Unforgeability of M-HSC scheme is based on Discrete Logarithm (DL) Problem, and specific analysis is as follows.

Theorem 2. In the random oracle, if an forger *F* has non-negligible advantage against the EUF-M-HSC-CMA security of our scheme when running in time *t* and performing q_s signcryption queries and q_i(i =1,2,3) hash queries ,then there is an algorithm *B* that solves the DL problem with probability $\varepsilon' \ge \varepsilon \cdot \frac{q_s(2q_2+2q_3+q_s-1)}{2q}$ within running time $t' \le t+$ $(t_f+q_s)O(t_1)$where t_1 denotes the time required for one discrete logarithm operation.

Proof. We will show how to build an algorithm *B* to solve the DL problem by running the forger *F* as a subroutine. On inputting(g^a, y), $y \in G_1$, the goal of *B* is looking for *a* ($0 \le a \le q, g^a = y$).

Initialization: *B* sends system parameters to *F* and generates *n* public / private key pairs (pk_s, sk_s), $(pk_{r1}, sk_{r1}), (pk_{r2}, sk_{r2}), ..., (pk_{rn}, sk_{rn})$,where $sk_s, sk_{ri} \in Z_p^*$, $pk_s = g^{sk_s}, pk_{ri} = g^{sk_{ri}} \in Z_p^*$, （ *i=1,2,……,n* ）. *B* returns $(pk_s, pk_{r1}, ..., pk_{rn}, sk_{r1}, ..., sk_{rn})$ to *F*.

Attack: *F* performs some polynomial bounded hash queries and signcryption queries. *B* can simulate the challenger to execute each phase of the EUF-M-HSC-CMA game for *F* as follows:

Let L_1, L_2, L_3 be used to store the results of the queries H_1, H_2, H_3 respectively, where h_1, h_2, h_3 are random oracles controlled by *B*.

H_1 queries: Inputs a public key pk_{ri} $(1 \le i \le n)$ to H_1, *B* checks if there exists (x_i,n_i) and $n_i = i$ in L_1. If such a tuple is found, *B* answers x_i; otherwise *B* randomly selects $\eta \in Z^*$, computes $x_i = h_1(pk_{ri} \cdot \eta \bmod p)$ and puts (x_i,i) into L_1, and returns x_i as the answer.

H_2 queries: *B* checks if there exists (*y*,*K*) in L_2. If such a tuple is found, *B* answers *K*; otherwise *B* randomly selects $K \in \kappa$, puts (*y*,*K*) into L_2 and returns *K* to *F*.

H_3 queries: *B* checks if there exists (*M*,*y*,*v*) in L_3. If such a tuple is found, *B* answers *v*; otherwise *B* computes $z \equiv y \bmod p$, submits (*M*,*z*) to H_3 oracle, and then puts(*M*,*z*,*v*) into L_3 and returns *v* to *F*.

Signcryption queries: *F* produces messages $(m_1, m_2, ..., m_n) \in G_1$, *B* first randomly chooses $r \in Z_p^*$, computes $\delta = g^r \bmod p$, runs the H_1 simulation process to obtain x_i $(i=1,2,......,n)$ and computes $M = (m_1 \oplus x_1 \| ...... \| m_n \oplus x_n)$. *M* is submitted to the signcryption oracle for obtaining (v, s, t). *B* computes $z = (pk_s g^v)^s \bmod p$, $C = Enc_K(M)$, puts (*z*,*K*) into L_2 and puts(*M*,*z*,*v*) into L_3, and returns $\sigma = (C, v, s, t, \delta)$ to *F*.

Forge: *F* produces a ciphertext σ^* and gives an arbitrary sender's public key pk_u.The σ^* is a valid ciphertext if the result of $Unsigncrypt(\sigma^*, pk_u, sk_{ri})(1 \le i \le n)$ is not "$\perp$". In the meanwhile, *F* can't do $Signcrypt(m^*, sk_u, pk_{r1}, pk_{r2}, ..., pk_{rn})$.

Analysis:

The case which σ^* is a valid ciphertext and indicates that *B* knows $g^a = g^{(sk_s+v)\cdot s} = (pk_s \cdot g^v)^s = y$, in other words, $a = (sk_s + v) \cdot s$, which is inconsistent with the assumptions.

In the game, the only thing that might fail is querying the values of H_2 and H_3 in signcryption queries. Because *F* does a maximum of q_2 H_2-queries and q_3 H_3-queries, possible number of different *y* is stored at most is $q_2 + q_3$.In the *i* signcryption query, *y* inconsistent probability is at most $\frac{q_2+q_3+(i-1)}{q}$. *F* runs q_s-times signcryption queries as far as possible, so the probability of *F* success is $\varepsilon' \ge \varepsilon \cdot \frac{q_s(2q_2+2q_3+q_s-1)}{2q}$.

6.3 Efficiency Analysis

When the sender sends *n* messages to *n* receivers, the length of the ciphertext in our scheme is $|nm| + 4|G_1|$. The length of the ciphertext is $n(|m| + (n+2)|G_1|)$of Ref.[4], which is larger than the length of the proposed scheme in this paper. In Ref.[16], the signcryption process requires three multiplication operations and its length of ciphertext is $(|nm| + 2|G_1| + n|G_2|)$. Therefore, it is longer than the one of this paper and does not facilitate transmission. In conclusion, compared with the existing schemes, the ciphertext of the proposed scheme is shorter. In our scheme, signcryption operation requires 0 pairing operation, *n* multiplications and (*n*+2) hash operations, but unsigncryption operation requires 0 pairing operation, *2* multiplications and 3 hash operations for a single receiver. Comparing with previous schemes, the efficiency of this scheme is better.

Table 1 compares M-HSC with schemes of Ref.[4], Ref.[15] and Ref.[16] in computational costs and communication overheads, where $|G_1|$ indicates the length of the element in the G_1, $|G_2|$ indicates the length of the element in the G_2, |m| indicates the length of the plaintext message *m*.

Table 1: Efficiency comparison between M-HSC and other schemes

Scheme	Signcryption (*n* receivers, *n* messages)			Unsigncryption (single receiver)			Ciphertext size
	Pair	Mul	Hash	Pair	Mul	Hash	
Ref.[4]	0	$n(n+2)$	$n(n+2)$	$2n$	n	$3n$	$n(\|m\|+(n+2)\|G_1\|)$
Ref.[15]	0	n	$n(n+2)$	0	n	$3n$	$n(\|m\|+3\|G_1\|)$
Ref.[16]	n	3	$3n+1$	1	3	3	$(\|nm\|+2\|G_1\|+n\|G_2\|)$
M-HSC	0	n	$n+2$	0	2	3	$\|nm\|+4\|G_1\|$

7. The Application of M-HSC in CATV Networks

With the wide use of the CATV, the security issues in the network become increasingly prominent. The business and consumers are very concerned about the topic how to establish a safe, convenient environment of CATV network and provide adequate protection to the user. We present a new broadcast service protocol using M-HSC scheme, which not only can effectively broadcast information, but also can prevent the possibility of fraud and destructive behavior. The protocol consists of three phases： system initialization, broadcast service and service certification.

System initialization. The on-line or off-line KGC for CATV networks generates system parameters $params=(G_1,g,q,p,h_1,h_2,h_3)$, and operators' public/private key pair (pk_s,sk_s) and users' public/private key pairs $(pk_{r1},sk_{r1}),(pk_{r2},sk_{r2}),......,(pk_{rn},sk_{rn})$ as described in the Section 5.

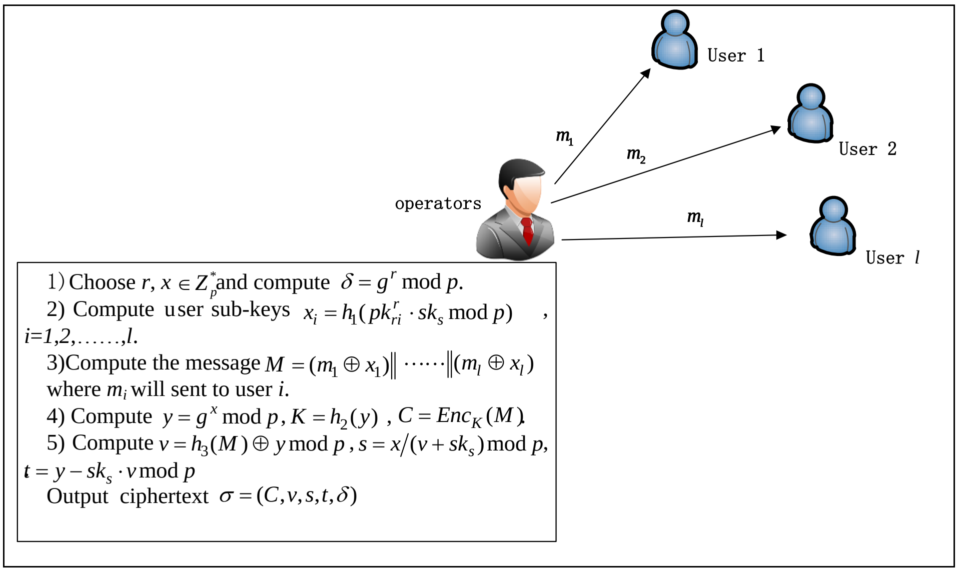
Broadcast Service. Each operator can simultaneously provide a number of different services for different users. Before broadcasting messages, the operator knows the users' public key, and the user has paid related fees. The procedure of broadcasting services is depicted in Fig.1.

Step1: The operators *A* determines the set of users $\{R_1,R_2,......,R_l\}$ and the set of services $\{m_1,m_2,......,m_l\},l\le n$, where *n* is the total number of users.

Step 2: *A* generates ciphertext σ as described in Section 5.

Step3: *A* sends σ to $\{R_1,R_2,......,R_l\}$ via a secure channel.

Service Certification. The user can authenticate the received broadcast messages by employing our scheme. The user can verify whether the received message is correct and intact. Fig.2 shows how a legitimate user obtains services provided by the operator.



1) Choose $r, x \in Z_p^*$ and compute $\delta = g^r \bmod p$.

2) Compute user sub-keys $x_i = h_1(pk_{ri}^r \cdot sk_s \bmod p)$, $i=1,2,\ldots\ldots,l$.

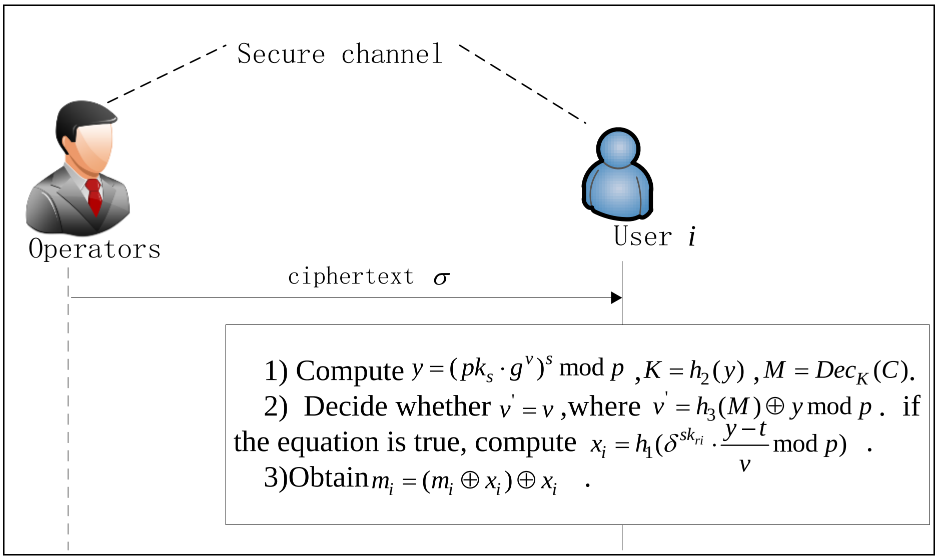
3)Compute the message $M = (m_1 \oplus x_1) \| \cdots\cdots \| (m_l \oplus x_l)$ where m_i will sent to user i.

4) Compute $y = g^x \bmod p$, $K = h_2(y)$, $C = Enc_K(M)$.

5) Compute $v = h_3(M) \oplus y \bmod p$, $s = x/(v + sk_s) \bmod p$, $t = y - sk_s \cdot v \bmod p$

Output ciphertext $\sigma = (C, v, s, t, \delta)$

Figure 1: Broadcast services of operators



1) Compute $y = (pk_s \cdot g^v)^s \bmod p$,$K = h_2(y)$,$M = Dec_K(C)$.

2) Decide whether $v' = v$,where $v' = h_3(M) \oplus y \bmod p$. if the equation is true, compute $x_i = h_1(\delta^{sk_{ri}} \cdot \frac{y-t}{v} \bmod p)$.

3)Obtain $m_i = (m_i \oplus x_i) \oplus x_i$.

Figure 2: Obtain service of legitimate user

8. Conclusions

In this paper, based on the discrete logarithm and GDH problem, we present a multi-message and multi-receiver hybrid signcryption scheme. Under the random oracle model, the formal demonstration shows that the proposed scheme can meet the indistinguishability of multi-message and multi-receiver hybrid signcryption, chosen ciphertext attack (IND-M-HSC-CCA2) and existentially unforgeable against chosen message attack (EUF-M-HSC-CMA). The analysis shows that our scheme not only is secure, reliable and verifiable, but also meets the fairness of decryption to prevent possible cheating behavior of the sender effectively. At the same time, the scheme can meet the requirement of businessmen where one signcryption operation will broadcast a number of different services to multiple receivers in the CATV networks environment.

References

[1]. Zheng Y. “Digital signcryption or how to achievecost (signature&encryption)<<cost(signature)+cost (encryption),” Advances in Cryptology—CRYPTO'97. Springer Berlin Heidelberg, pp.165-179, 1997.

[2]. J.H.Koo, H.J.Kim, I.R.Jeong. “Jointly unsigncryptable signcryption schemes,” Proceedings of WISA. Vol.1, pp.397-407, 2001.

[3]. Seo M, Kim K. “Electronic funds transfer protocol using domain-verifiable signcryption scheme,” Information Security and Cryptology-ICISC’99. SpringerBerlin Heidelberg, pp.269-277, 2000.

[4]. F G Li. “signcrypted system research based on bilinear pairings,” Ph.D. thesis, Xidian University, 2007.

[5]. Cramer R,Shoup V. “Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack,” SIAM Journal on Computing,Vol.33,no.1, pp.167 – 226, 2003.

[6]. Abe M, Gennaro R, and Kurosawa K. “Tag-KEM/DEM: a new framework for hybrid encryption,” Journal of Cryptology, Vol.21, no.1, pp.97–130, 2008.

[7]. Kiltz E. “Chosen ciphertext secure key encapsulation based on gap hashed diffie-hellman,” In Public Key Cryptography-PKC 2007, LNCS 4450. Berlin:Springer-Verlag, pp.282–297, 2007.

[8]. Http: //eprint.iacr.org/2004/210/, accessed December 2014.

[9]. Bjørstad T E, Dent A W. “Building better signcryption schemes with tag-KEMs,” Public Key Cryptography-PKC2006. Springer Berlin Heidelberg, pp.491-507, 2006.

[10]. Tan, Chik How. “Insider-secure Hybrid Signcryption Scheme Without Random Oracles,” Availability, Reliability and Security, pp.1148-1154, 2007.

[11]. Li Fagen, Shirase M, Takagi T. “Identity-based hybrid signcryption,” Availability, Reliability and Security, 2009.(ARES'09). International Conference on. Fukuoka, Japan, pp.534-539, 2009.

[12]. Y Ning. “Application of digital broadband integrated information network based on CATV,” Network Technology,Vol.7, pp.45-47, 2012.

[13]. Smart N P. “Efficient key encapsulation to multiple parties,” Security in Communication Networks. Springer Berlin Heidelberg, pp.208-219, 2005.

[14]. Sun Y, Li H. “ID-based signcryption KEM to multiple recipients,” Chinese Journal of Electronics, Vol.20, no.2, pp.317-322, 2011.

[15]. R F li. “Design and realization of a hybrid signcryption,” Ph.D thesis, Xidian University, 2013.

[16]. Jing Q, Jun B, Xin X S, Su M H. “Secure and efficient multi-message and multi-receiver ID-based signcryption for rekeying in ad hoc networks,” Journal of Chongqing University(English Edition)[ISSN 1671-8224], Vol.12, no.2, pp.91-96, 2013.

Caifen Wang received the MS degree in mathematics from Lanzhou university, China, in 1998,and the PhD degree in cryptography from Xidian university in 2003.she is a professor in Northwest normal university, she also is a visiting professor of department of electronic engineering in Chin-Yi University of Technology ,Taiwan . Her current researcher interests including information security, cryptographic protocols and electronic commerce.