Modeling of Multiple Agent based Cryptographic Key Recovery Protocol

Shinyoung Lim, Sangseung Kang, Joochan Sohn

ETRI, Korea

{limsy, ssk, jcsohn} @etri.re.kr

Abstract

When a receiver of a ciphertext message can not decrypt the message due to the fact that he has lost his private-key, the private-key of the receiver and session-key of the message need to be recovered. In this paper, we demonstrate how we have modeled and analyzed a new type of multiple agent based key recovery protocol. It is characterized by key encapsulation approach, protocol generalization, secret choice of key recovery agents and fork/join of session-keys by random-keys. The proposed protocol is formally modeled by a new pictorial model, an Extended Cryptographic Timed Petri Net (ECTPN). Recoverability of a session-key as well as performance of the protocol is verified by using a reachability graph of the ECTPN.

1. Introduction

Key management, like cryptographic algorithms, is a core technology in information technology. Key management is the group of activities that involves generation, distribution, validation, storage, usage, expiration, and recovery of cryptographic keys [1]. With regard to key recovery, there are several conflicts between preservation of personal privacy and profit of a government because personal secrecy may be violated by means of the key recovery.

The *key recovery* techniques, techniques for crypto- graphic information recovery, are classified by the following approaches [1,2]:

- **Key escrow** (e.g., Clipper chip or Escrowed encryption standard [3,4,5]): The user sends the cryptographic key to one or more escrow agents. When a user needs to recover the key, the escrow agent provides the stored key to the user.
- **Trusted third party** (e.g., Yaksha system[6,7], ANSI X9.17): The user gets their session-key from a trusted third party (e.g., key distribution center). When the user needs to recover the key, the trusted third party provides the original cryptographic key to the user.
- **Commercial key backup** (e.g., AT&T Crypto- Backup [8]): When data is encrypted, a session-key is encrypted using the public-key and kept with the data. The user sends the private-key to a key recovery agent. When a user needs to recover the key, the key recovery agent provides the stored private-key to the user.
- **Key encapsulation** (e.g., TIS RecoveryKey [9], CyKey [10], SecretAgent [11], IBM SKR [12], Binding Crypto-graphy): A session-key is encrypted along with other key recovery information in a capsule (i.e., key recovery block) which only the user's designated key recovery agent can decrypt. When the user needs to recover the key, the key recovery block is sent to the key recovery agent who then returns the recovered session-key to the user.

Key distribution center and escrow agent are a bottleneck of the information technology infrastructure with high transaction overhead and large storage requirements. Private-key backup reduces the overhead, but many users do not want to submit their private-key to any kinds of authority since it provides access to all of their encrypted data. The *key escrow* approach is sensitive to violation of privacy, and suffers from a binding problem [5](i.e., possibility of secret communication among users without monitoring by the government). The *key encapsulation*, however, is relatively free from violation of privacy (because the owner of a key has the authority of key recovery) and has many advantages over other approaches.

However, conventional *key encapsulation* approaches have the following problems [9,10,11]. First, they lack a formal modeling and analysis model as well as implementation technology which are useful for real developers of key recovery systems. Second, a user agent who is a requester of key recovery, needs to directly communicate with one or more key recovery agents in some approaches [9, 10]. Thus, all user agents must have complex functions of the key recovery. Finally, there is no practical proposition concerning the multiple agent based key recovery system that consists of multiple key recovery agents.

Coping with these problems, we have developed a new multiple agent based key recovery protocol, and proposed a new analysis model, an Extended Cryptographic Timed Petri Net (ECTPN), for the purpose of formal modeling and analysis of the protocol. Note that the *Petri net* model has been successfully used as a formal and pictorial modeling and analysis model for concurrent and real time systems since 1964 [13,14]. The Petri net also has been used for modeling and analysis of information flows [15] and cryptographic protocol [16] of information technology security systems.

This paper is organized as follows: Section 2 describes requirements of the key recovery and framework of the protocol. Section 3 presents a formal modeling and results of analysis of the protocol by means of an ECTPN model and its reachability graph. Basic definition of the ECTPN is presented in Appendix. Section 4 describes and compares conventional key encapsulation approaches and our approach. Finally, we conclude our approach in section 5.

2. Framework of Research

2.1 Problems and requirements of key recovery

Assuming a user-A (UA) sends a ciphertext message to a user-B (UB) by means of the cryptographic mechanism over a public key infrastructure, and if a UB has lost his private-key or a UA has been encrypted by a UB's old public-key, then the *session-key* can not be obtained and the ciphertext data can not be decrypted. Thus, a UB or a law enforcement agent should recover the session-key. Basic requirements and assumptions about the key recovery protocol are listed as follows:

- :y recovery should be done under the control of end users or security policy of an organization. In this context, we may not solve the *binding* problem[8]. Thus a law enforcement access field (LEAF) is not included in encapsulated key recovery information that is encapsulated in a communication message.
- rget of recovery is restricted to the *session-key* (not the data) which is a secret-key for encryption of data. Thus, we can use the *key encapsulation approach* because the size of a session-key is relatively smaller than that of the data.
- solution should be formally specified and validated, as well as performance of the solution should be evaluated.
- ıe key recovery protocol is executed over a conventional public key infrastructure. Thus, we assume that the public-keys and certificates of all agents in the key recovery protocol have been distributed securely by a key distribution center or certification authority over a public key infrastructure.

2.2 Framework of proposed multiple agent based key recovery protocol

The proposed protocol is executed over key recovery systems. The key recovery system is consisted of nested security domain as shown in Fig. 1. Note that key recovery provider domain is the most secure domain in the key recovery system. The key recovery service is only provided to user agents in the key recovery domain. i.e., intra-key-recovery communication. The key recovery scenario of the multiple agent based key recovery protocol is presented as follows:

- **Phase 1 (*initialization*)**: Public-keys of all agents in the application domain are distributed by a public key infrastructure after certification authority issued certificates for agents. The certificate contains public-key of agents and information for identification and authentification.
- **Phase 2 (*communication and key recovery information generation*)**: A UA *secretly* or *randomly* chooses one or more key recovery agents among a pool of key recovery agents and generates key recovery information (KRI) by using public-keys of KRC and the chosen key recovery agent. The KRI is encapsulated within communication message along with the ciphertext of the data. Thus, the message is constituted and sent to a UB. A UB, however, can not decrypt the ciphertext message because he has lost his private-key (prk_B) or the message has been encrypted by his expired public-key. Note that a UB can not know which

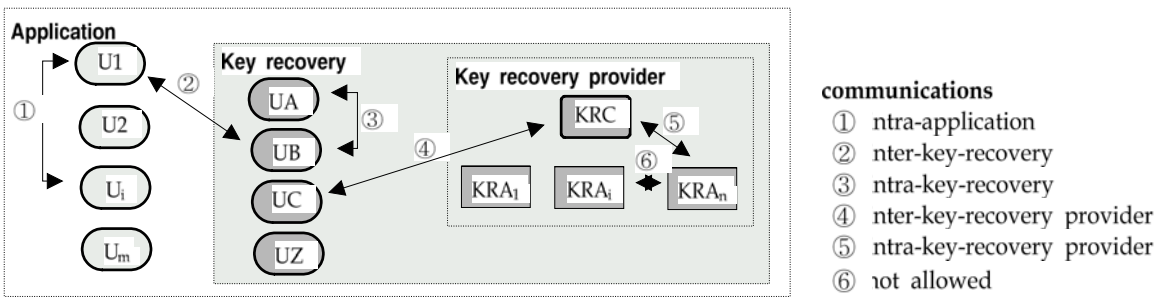


Fig. 1. Domain of the multiple agent based key recovery system

key recovery agents are chosen by a UA. And it is also possible when an information system manager of the organization has found that a UA violated the security policy of the organization (or government).

- **Phase 3 (*key recovery*)**: A UB or a law-enforced third party, in this case, the information system manager, who has the recovering message, generates a new key pair (puk_B', prk_B'), and registers the puk_B' to certification authority. The certification authority gives a certificate $cert_B$, including the new registered key puk_B', to a UB. A UB officially requests key recovery service to the KRC by sending his $cert_B$, req_B and KRI. Thus, the key recovery process is proceeded according to ④ ⑧ of Fig. 2.

The multiple agent based key recovery protocol means that a UA chooses *n* KRAs from a pool of KRAs in key recovery provider domain. For example, KRA1 and KRA2 are involved in the 2 agent based key recovery protocol. A session-key, *ssk,* that is to be recovered by the protocol, is forked into *n* intermediate-keys, ik_i, at UA, and joined them into a recovered key at KRC. The fork and join function are defined as follows:

- fork(ssk, n) = ik_1, ... ik_i, ... ik_n (where, $ik_1 = ssk \oplus rk_1$, $ik_2 = rk_1 \oplus rk_2$, ..., $ik_i = rk_{i-1} \oplus rk_i$, ..., $ik_n = rk_{n-1}$),
- join(ik_1, ... ik_i, ... ik_n) = $ik_1 \oplus ik_2 \oplus$...., $\oplus ik_i \oplus$ ... $\oplus$ ik_n, (rk_i is a random-key).

Note that the *ssk* is preserved because $(\boldsymbol{ssk} \oplus rk_1) \oplus (rk_1 \oplus rk_2) \oplus$...., $(rk_{i-1} \oplus rk_i)$, ... $\oplus$ $(rk_{n-2} \oplus rk_{n-1}) \oplus rk_{n-1} = \boldsymbol{ssk}$.

An ik_i is encrypted at UA and decrypted at KRA_i by public-key and private-key of KRA_i. Intermediate-keys are merged at UA and divided at KRC in order to constitute KRI and communication message.

3. Modeling and Analysis of *multiple agent based* Key Recovery Protocol

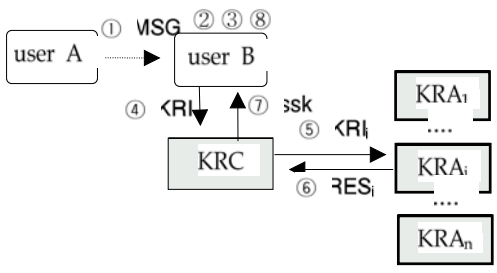
The basic definition of an Extended Cryptographic Timed Petri Net (ECTPN), modeling tool for multiple agent based key recovery protocol, is presented in Appendix. Modeling activity means constructing of an ECTPN structure for some protocol. In other word, an ECTPN structure is a pictorial and formal specification of the multiple agent based key recovery protocol.

Key, message and data in a protocol are modeled by *places* of an ECTPN. Functional components (e.g., cryptographic, operation and communication functions) and predefined high-level functional components in a protocol are modeled by *transitions*. Availability or activation of data is specified by a *token* in corresponding place. An ECTPN model for the multiple agent based key recovery protocol is constructed by means of stepwise refinement (top-down), incremental (bottom- up) or mixed approach.

3.1 Modeling by bottom-up approach

(1) Modeling of key, message and data by places

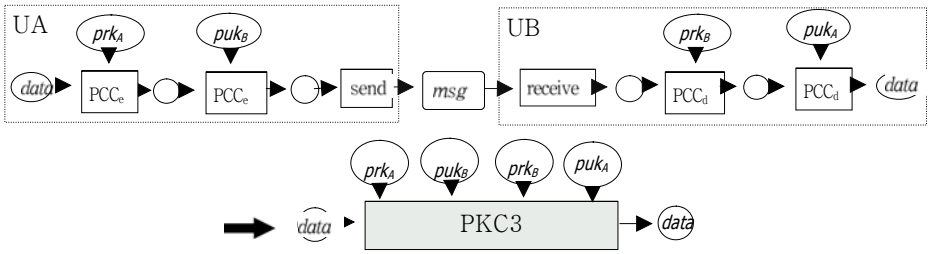
- *Key places*: *ssk* (session-key), *prk* (private-key), *prk'* (a new private-key for key recovery), *puk* (public-key), *puk*' (a new public-key for key recovery), *rk* (random- key for Vernam cipher), *ik* (intermediate-keys for *ssk*), *s-prk* (private-key for signature), *cert* (certificate), *sig* (signature), *s-puk* (public-key for verification), *krr* (key recovery request)
- *Message places*: *msg* (message), *KRI* (key recover information)
- *Data places*: *data* (any data), *req* (key recovery request), *int*(integrity) (Note: For each place p_i, size of it, *size* (p_i), is annotated for the purpose of performance evaluation.)



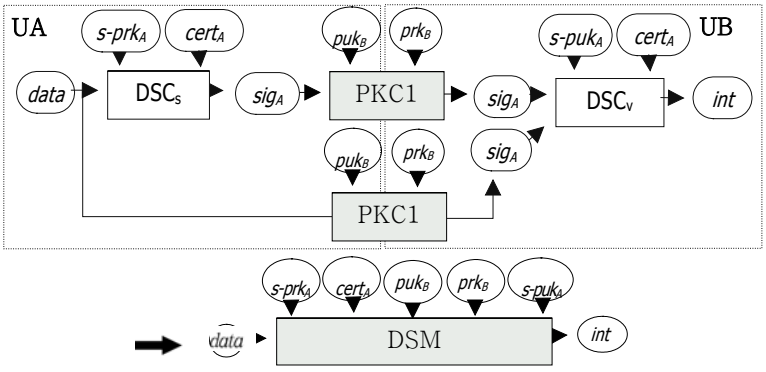
① MSG = SC(*data, ssk*) + KRI, where, KRI = merge(KRI_i = PC(PC(fork(*ssk, i*), puk_{KRAi}), puk_{KRC}))
② ③ prk_B is lost or changed; Generates new prk_B' and puk_B' and registers the puk_B' at certification authority; Request key recovery to KRC (by UB's request or law enforcement)
④ Request key recovery to KRC with KRI
⑤ KRI = PC(KRI, prk_{KRC}), KRI_i = divide(KRI), Request key recovery to KRA_i with KRI_i
⑥ RES_i = PC(KRI_i, prk_{KRAi}) ⑦ ssk = join(RES_i)
⑧ MSG = SC(*data, ssk*)

Note: SC(x,y) : a Secret-key Cryptographic function such as DES (x is data, y is key)
PC(x,y) : a Public-key Cryptographic function such as RSA

Fig. 2. Framework of the multiple agent based key recovery protocol



(a) PKC3(two agent based public key cryptographic macro transition)



(b) DSM(a digital signature macro)

Fig. 3. Examples of macro transitions

(2) Bottom-up modeling of component by transitions

(a) modeling of functional components

Functional components in the protocol are modeled by the following types of component transitions.

- *Operation component transitions*: *merge* (concatenation), *divide* (partition), ⊕(exclusive-or), =?(equality check)
- *Communication component transitions*: *send* (send function), *receive* (receive function) (Note: When key recovery is needed for a stored encrypted data, *send/receive* transitions are replaced by *write/read* transitions, respectively.)
- *Cryptographic component transitions*:

(Remarks: An element of set is an instance of component, $*_e$ (encryption), $*_d$ (decryption), $*_s$ (signature), $*_v$ (verification))

- Secret-key cryptographic component (SCC) = {DES}

  SCC_e (*data, session-key*), SCC_d (*data, session-key*)

  SCC is identical to symmetrical key cryptographic component.

- Public-key Cryptographic Component (PCC) = {RSA}

  PCC_e (*data, public-key*), PCC_d (*data, private-key*)

  PCC is asymmetrical key cryptographic component. Every agent has a pair of key, a public-key and a private-key. Distribution of the public-key should be preceded because a sender encrypts a data by the public-key of receiver.

- Hash Component (HAC) = {SHA}

  HAC (*data*)

  HAC is composed of one way function, namely irreversible one way function. It is used for digesting a certain data before digital signature.

- Digital Signature Component (DSC) = {KCDSA}

  DSC_s (*data, signature-private-key, certificate*)

  DSC_v (*data, data-signature, signature-public-key, certificate*)

  DSC is composed of signing data digitally and verifying digital signed data. In most case, the input data is hashed data, but, in case of KCDSA (Korean Certification-based Digital Signature Algorithm [17]), the input data is data itself because hash component is included in KCDSA.

(b) modeling of macro

A *macro* is high-level functional module in a protocol. *Macro transitions* are used for the purpose of high level modeling of system. Some useful macro transitions are listed as follows ("X ::= Y" means "X is defined by Y"):

- Secret-key cryptographic macro transition (SKC)

  SKC (*data, ssk*) ::= SCC_d (*receive* (A, *send* (B, SCC_e(*data, ssk*))), *ssk*)

- One way public-key cryptographic macro transitions (PKC1, PKC2)

  PKC1 (*data, puk_B, prk_B*) ::= PCC_d (*receive*(A, *send*(B, PCC_e (*data, puk_B*))), *prk_B*)

  PKC2 (*data, prk_A, puk_A*) ::= PCC_d (*receive*(A, *send*(B,

PCC_e (*data*, prk_A))), puk_A)

- Two way public-key cryptographic macro transition (PKC3)

  PKC3 (*data*, prk_A, puk_A, prk_B, puk_B) ::= PCC_d(PCC_d (*receive*(A, *send*(B,PCC_e(PCC_e(*data*, prk_A), puk_B))), prk_B), puk_A)

- Integrity and secret cryptographic macro transition (ISC)

  ISC(*data*, *ssk*, prk_A, puk_A, prk_B, puk_B) ::=
  if PKC2(HAC(*data*),prk_A,puk_A) = HAC(SKC(*data*, PKC3(*ssk*, prk_A, puk_A, prk_B, puk_B)))
  then *data* **else** error.

- gital signature macro transition (DSM)

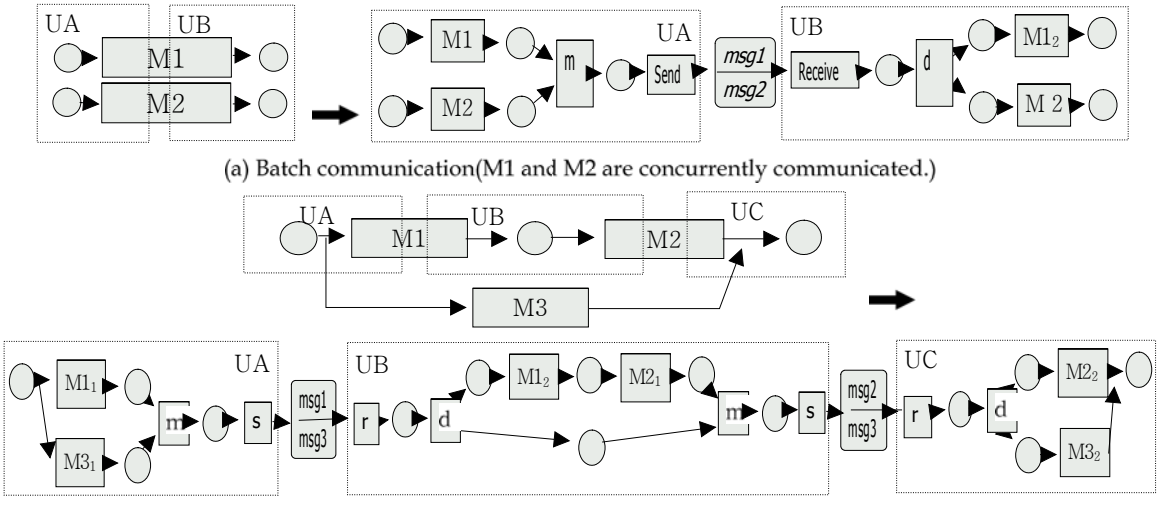
  SIG(*data*, *s-*prk_A, $cert_A$, puk_B, prk_B, *s-*puk_A) ::=
  DSC_v(PKC1(DSC_s(*data*, *s-*prk_A, $cert_A$), puk_B, prk_B), PKC1(*data*, puk_B, prk_B), *s-*puk_A, $cert_A$)

Some macro transitions are shown in Fig. 3. *Component transitions* are implemented by using *components* in conventional cryptographic and communication libraries. *Macro transitions* are the high level crypto- graphic and communication application programming interfaces that is the same concept as open data base connectivity in a database application. Macro transitions can be implemented using the rule in Fig. 4.

delay(comm)+ *delay*(PCC_d) +*delay*(PCC_d), where 'comm' denotes communication components.

Table 1 presents some annotated delay value of cryptographic and communication transitions. These values are modified depending on computation and communication infrastructure of agents in the key recovery system.

Table 1. Annotated delay value of some component transitions



(a) Batch communication(M1 and M2 are concurrently communicated.)

(b) Transitive communication

Note: Mi is a macro transition. Mi_1 and Mi_2 are partial computations(e.g., RSA, DES) within a sender and a receiver, respectively. m, d, s and r denote merge, divide, send and receive transition, respectively.

Fig. 4. Implementation rules for macro transitions

(3) Annotation of delay and reliability to transitions

When the performance of the multiple agent based key recovery protocol should be evaluated, execution delay values *delay*(t_i) are annotated to each component transitions t_i. The performance of a macro transition M is computed as follows: *delay*(M) = $\sum$*delay*(t_i), where t_i is an elementary transition of M.

For example, *delay*(PKC3) = *delay*(PCC_e)+*delay*(PCC_e)+

transitions	delav (ms)(*)	data size(bytes)		
		input	key	output
DES_e	.440	300	16	300
DES_d	.128	300	16	300
RSA_e	1.635	32	97	64
RSA_d	10.331	64	344	32
RSA_s	10.263	20	344	64
RSA_v	1.332	64	97	20
SHA	.382	97	-	20
$KCDSA_s$	50.643	289	230	120
$KCDSA_v$	85.126	289	180	1 (logic)
communication transitions(**)		size of message		
UA → JB	592.02	4140		
UB → JRC	659.66	4613		
KRC → JRA1. KRA2	76.93	538		
KRA1 → JRC , KRA2 → KRC, KRC → JB	9.15	64		

(*) measured on SUN Ultra 10, 333 MHz, 256 MB
(**) measured on 7KByte/Sec bandwidth

(4) Result of ECTPN model of 2 agent based key recovery protocol

A high-level ECTPN modeling of the 2 agent based key recovery protocol is presented in Fig. 5. Note that the ECTPN model of the protocol can be regarded as an application program interface of the protocol because the protocol was developed using components in cryptographic and communication class libraries.

(5) Implementation of prototype 2 agent based key recovery system

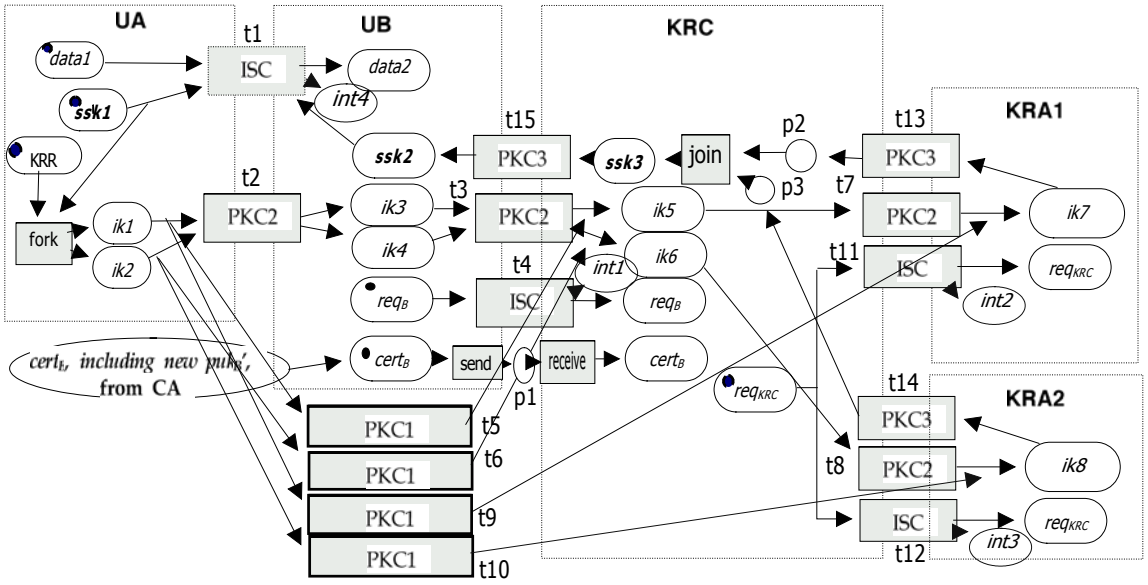
We have implemented a prototype key recovery system by means of the component based software engineering technology. The cryptographic components such as SCC, PCC, HAC and DSC are implemented by cryptographic algorithms such as DES, RSA, SHA and KCDSA, respectively.

3.2 Analysis

(1) Recoverability analysis

The *recoverability* of *ssk* can be verified by means of *reachability* analysis of an ECTPN. For example, if *ssk2* place in a UB is reachable from initial marking of a UA, then *ssk1* is recoverable from the protocol. Thus, we can show that all transitions in the protocol are reachable from an initial state by using a reachability graph.

A *Reachability Graph (*RG) is defined as follow: RG = <N, LA>, where, N is a set of nodes, and LA is a set of labeled arcs connecting two nodes. Note that a node is a set of concurrently marked places, and represents a state of an ECTPN. A label of LA represents concurrently fired transitions.



· ssertions: $data1 = data2$, $ssk1 = ssk2 = ssk3$, $ik1 = (ssk1 \oplus rk)$, $ik2 = rk$, $ik3 = PCC_e(PCC_e(ik1, puk_{KRA1}, puk_{KRC}), ik4 = PCC_e(PCC_e(ik2, puk_{KRA2}), puk_{KRC}), ik5 = PCC_e(ik1, puk_{KRA1}), ik6 = PCC_e(ik2, puk_{KRA2}), ik7 = ik1, ik8 = ik2, ssk2 = ik7 \oplus ik8, int1 = int2 = int3 = int4 =$ true

· ote: Distribution procedure of public-keys from CA is omitted.

Fig.5. High-level ECTPN model of 2 agent based key recovery protocol

As an example, a label $t_i(p_{i1}, ...p_{ik}....) \parallel t_j(p_{j1}, ...p_{jk}....)$ means that t_i and t_j have finished concurrent firing, and p_{ik} and p_{jk} are output place from t_i and t_j, respectively. Firing the enabled transition in N_i, a new node N_j is generated after the total delay which is annotated to the enabled transitions as shown below:

LA_{ij} [$t_1(p_{11}, ...p_{1i}....) \parallel t_2(p_{21}, ...p_{2k}....) ..... \parallel t_n(p_{n1}, ...p_{nk}....)$] : $N_i \rightarrow$? after ($delay(t1)$+ .... + $delay(t_i)$ + ... +$delay(t_n)$). $p_{i1}, ..., p_{nk} \in$?$_i = \{mp_{i1}, mp_{i2}, ... mp_{ik}\}$

where, mp is marked place and $delay(t_i)$ is annotated delay time of t_i.

A reachability graph of an ECTPN of 2 agent based key recovery protocol is presented in Fig. 6.

(2) Recovery scenario and delay

A recovery scenario SC_i is a path over reachability graph. Let SC_i = { N_{i1}, LA_{i1}, ..., N_{ik}, LA_{ik}, ..., N_{im} } (where, LA_{ik} is an arc from N_{ik} to N_{ik+1}). Total delay of SC_i, $delay(SC_i)$, is computed as follows:

$$delay(SC_i) = \sum_{k=1}^{m} delay(LA_{ik}), \text{ where } delay(LA_{ik}) = (delay(t1)+...+delay(t_i)+ ... +delay(t_n)), \ t_i \in A_{ik}.$$

(3) Performance

The *performance* refers to the time consumed for key recovery operation. We assume that all KRA have the same computational capability, and does not include additional service time such as encrypted data decryption time.

Let $LOT(i)^{(n)}$ and $CCT(X,Y)^{(n)}$ be *local computation time* of node *i* and *cryptographic communication time* from node X to node *Y*, respectively. $LOT^{(n)}$, $CCT^{(n)}$, $TRT^{(n)}$ and $PER^{(n)}$ be *computation time, communication time, total response time* and *relative performance factor* of the multiple agent based key recovery protocol, respectively.

- $LOT^{(n)} = LOT(\text{UA})^{(n)} + LOT(UB)^{(n)} + LOT(KRC)^{(n)} + LOT(KRA)^{(n)} \times n$
- $CCT^{(n)} = CCT(UA,UB)^{(n)} + CCT(UB,KRC)^{(n)} + CCT(KRC,UB)^{(n)} + [(CCT(KRC,KRA)^{(n)} + CCT(KRA,KRC)^{(n)}] \times n$
- $TRT^{(n)} = LOT^{(n)} + CCT^{(n)}$
- $PER^{(n)} = RT^{(1)} / RT^{(n)}$

4. Related Approach and Discussion

4.1 Other key encapsulation based approaches

In commercial key recovery (CKR)[9] and CyKey [10], key recovery service is provided by a data recovery center[9] or a key recovery requester which has sanctioned by key recovery authority[10]. Thus, all user agents, who request key recovery service, directly communicate with the recovery center. The recovery center not only is venerable to disguising or impersonation attacks, but also monopolizes whole authority of key recovery. Access rule index, a primary distinguishing characteristic of CKR[10], is a number indicating the access rule. Access rules are the processes for verifying permission to gain emergency access and thereby defines a set of people qualified to receive access, possibly subject to some conditions. In Secret Agent [11], recovery service is provided by a key recovery requester and a KRA as shown in Fig 7-(a). Thus, the security of key recovery is increased because at least two agents are involved in the key recovery service.

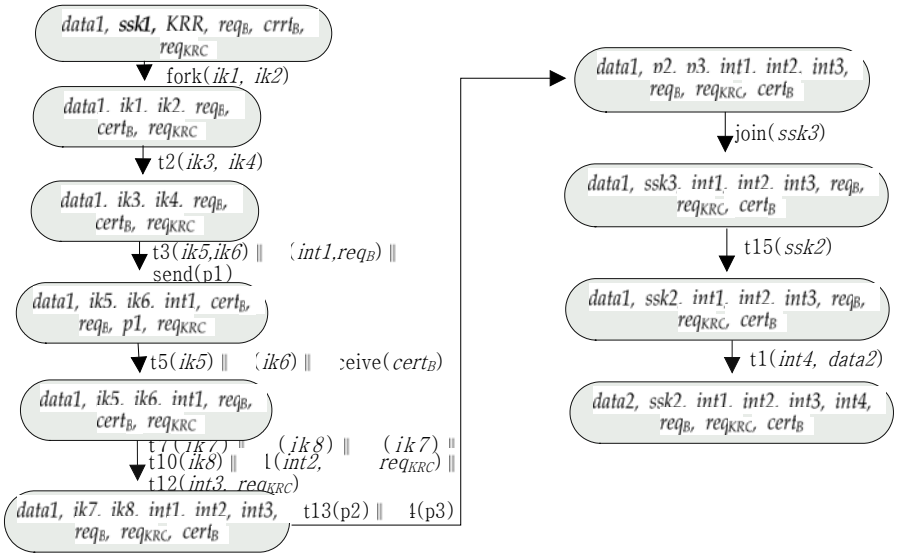


Fig. 6. Reachability graph of the high-level model of ECTPN in Fig. 6

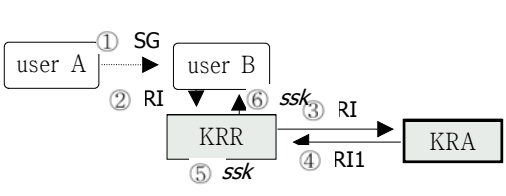
In the SKR (two-phased cryptographic secure key recovery) [12], the recovery service is provided by a recovery provider and two or more KRAs as shown in Fig. 7-(b). The SKR involves a costly public encryption operation to be performed one time by using key recovery information (KRI) which resulted from phase 1, and then used multiple times to generate the needed key-encrypting keys that will enable the recovery of multiple keys over a potentially long period of time by using the KRI that resulted from phase 2. Therefore, secret values provided to the key recovery agent are potentially long-lived key-generating (KG) keys, and the secret values used to cryptographically seal the keys to be protected by the SKR are key-encrypting (KK) keys. KK are derived from these KG using a key derivation procedure that tightly couples the derived KK to the unique public KRI associated with the key being protected by the SKR.

In the SKR, recoverability is analyzed by parameter validation scheme. The SKR, however, is suitable for a cryptographic application whose cryptographic sessions are easily aggregated into a session group. Additionally, a key recovery requester knows KRAs of a sender because KRAs are fixed. The SKR has complex procedure of the key generation.

4.2 Discussion

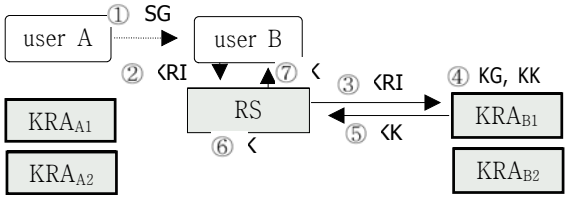
Comparing to conventional encapsulation-based key recovery protocols, distinguishing characteristics of the multiple agent based key recovery protocol are summarized as follows:

- **Secret choice of key recovery agents**: A sender of cryptographic message, who has a duty of generation of key recovery information, can secretly choose his key recovery agents among a pool of key recovery agents in the key recovery provider domain. A receiver of the message needs not know the identification and public-keys of the chosen key recovery agents. Thus the possibility of collusion among key recovery agents is reduced.
- **Fork/join of session-key by random-keys**: A 128 bits session-key is not physically divided into n intermediate-keys (i. e., each length is $128/n$ bits), which means exponentially degrade cryptographic complexity ($2^{-128/n}$), but n randomized 128 bits intermediate-keys are generated by exclusive-or operation of a session-key and $n-1$ random-keys. As a result, cryptographic complexity is preserved. A brute-force attacker needs to attack on the 128 bits session-key or n private-keys of each key recovery agents. The latter attacking is impossible without collusion with all of n key recovery agents.
- ıltiple agent based **key recovery approach**: The security and integrity of key recovery service is exponentially increased by means of increasing the numbers of key recovery agents. However, there is the trade-off problem between security and performance of the multiple agent based key recovery protocol. The proposed protocol can be directly applied to implementation of a key recovery system which has any numbers of key recovery agents.
- **Pictorial and formal modeling and analysis**: The multiple agent based key recovery protocol is modeled and analyzed by using an ECTPN model. The ECTPN model, including various modeling conventions such as cryptographic component transition and macro transition, is a useful model for any kind of cryptographic protocols.



① MSG = SC(*data, ssk*) + KRI
(where, KRF = PC(PC(*ssk*, puk_{KRR}), puk_{KRA})
②③ ⟨RI
④ ⟨RI1 = PC(KRI, prk_{KRA})
⑤ *ssk* = PC(KRI1, prk_{KRR})
⑥ *ssk*

(a) ISC's SecretAgent[11]



①② MSG = SC(*data. ssk*) + KRI
KRI = [B1 + B2]; KG = Hash(S
B1 = [T1, KG_{B1}' = PC(KG, puk_{B1}),
KG_{B2}' = PC(KG, puk_{B2})]: one time
B2 = [T2. Hash(B1), $Xb^{(i)}$ = SC(SC($K^{(i)}$, $KK^{(i)}_{B2}$),
$KK^{(i)}_{B1}$)]: per each session *i*
③ ⟨RI (by user B's request or law enforcement)
④ ⟨G_{B1} = PC(KG_{B1}'. prk_{B1}), KG_{B2} = PC(KG_{B2}'. prk_{B2}): one time
$KK^{(i)}_{B1}$ = Gen(KG_{B1}), $KK^{(i)}_{B2}$ = Gen(KG_{B2}): per each session *i*
⑤ ⟨$K^{(i)}_{B1}$, $KK^{(i)}_{B2}$
⑥⑦ ⟨$^{(i)}$ = SC(SC($Xb^{(i)}$, $KK^{(i)}_{B1}$), $KK^{(i)}_{B2}$): per each session *i*

(b) IBM's SKR[12]

Fig. 7. Conventional key encapsulated key recovery systems(Certification process among agents is omitted.)

- **Component based software engineering (CBSE) approach**: The CBSE technology, which is originated from CORBA of OMG, DCOM/OLE of Microsoft and JavaBean of Sun, is a state of art technology in information technology. We do not develop new cryptographic algorithms [19], but implement the multiple agent based key recovery system by using conventional secure cryptographic components in the crypto- graphic library and *macro* transitions. Thus the maintainability of the key recovery system is increased.

5. Conclusion

Key recovery is to be done not only when a private-key of a receiver has been lost, but also when a receiver has encrypted a message by using the *old* public-key of the receiver.

The multiple agent based key recovery protocol is a new key encapsulation based key recovery approach which is characterized by key encapsulation approach, protocol generalization, secret choice of key recovery agents, and fork/join of session-keys by random-keys. Additionally, an extended cryptographic timed Petri net is useful model for pictorial modeling, analysis and component based implementation of any kind of cryptographic protocols.

The multiple agent based key recovery system, executing over a public key infrastructure, can be used as many kinds of security solutions in Web based applications such as electronic commerce and electronic data interchange. Further research in security assurance evaluation and measurement of the system performance in various user environment are required in our approach.

References

[1] Technology Committee of Key Recovery Alliance, Cryptographic Information Recovery using Key Recovery, A Working Paper, Version 1.2, http:// www.kra.org, Aug. 1997.

[2] Dorothy E. Denning and Dennis K. Branstad, "A Taxonomy for Key Escrow Encryption Systems," *Communications of the ACM*, pp. 34-40, Vol.39, No.3, 1996.

[3] Ravi Ganesan, "How To Use Key Escrow," *Communications of the ACM*, pp. 33, Vol.39, No.3, 1996.

[4] Jingmin He and Ed Dawson, "A New Key Escrow Cryptosystem," *Lecture Notes in Computer Science*, Vol. 1029, pp. 105-113, 1995.

[5] Yung-Cheng Lee and Chi-Sung Laih, "On the key recovery of the Key Escrow System," *Proceedings of 13th Annual Computer Security Applications Conference,* pp. 216 -220, 1997.

[6] Ravi Ganesan, "The Yaksha Security System," *Communications of the ACM*, Vol.39, No.3, pp. 55-60, 1996.

[7] Jefferies, N., Mitchell, C. and Walker, M., "A Proposed Architecture for Trusted Third Party Services," *Lecture Notes in Computer Science*, Vol. 1029, pp. 98-104, 1995.

[8] D. P. Maher, "Crypto Backup and Key Escrow," *Communications of the ACM*, Vol. 39, No.3, pp. 48-53, 1996.

[9] Stephen T. Walker, Steven B. Lipner, Carl M. Ellison, and David M. Balenson, "Commercial Key Recovery," *Communications of the ACM*, Vol.39, No.3, pp. 41-47, 1996.

[10] M. Markowitz and R. Schlafly, "Key Recovery in SecretAgent," *Digital Signature*, 1997.

[11] Cylink Corp., "CyKey, A Key Recovery System for Commerical Environments," http://www.cylink.com, 1998.

[12] R. Gennaro, et. al., "Secure Key Recovery," IBM Thomas J. Watson Research Center, 1999.

[13] James Peterson, J., *Petri Nets Theory and the Modeling of Systems*, Prentice-Hall, 1982.

[14] Zuberak, W., "Timed Petri Nets: Definitions, Properties, and Applications," *Microelectronics and Reliability*, Vol. 31, pp. 627-644, 1991.

[15] Varadhajan, V., "Petri Net based Modeling of Information Flow Security Requirements," *Proc. of the Computer Security Foundations Workshop III*, pp. 51-61, 1990.

[16] Gang-Soo Lee and Jin-seok Lee, "Petri Net based models for specification and analysis of Cryptographic Protocols," *Journal of systems and software*, Vol. 37, pp. 141-159, 1997.

[17] Korea Information Security Agency, Korean Certification-based Digital Signature Algorithm, 1997.

[18] Pressman, R. S., *Software Engineering A Practitioner's Approach*, third edition, McGraw-Hill, 1992.

[19] B. Scheneier, *Applied Cryptography*, second edition, Wiley& Sons, 1996.

[20] Requirements for Key Recovery Products, Final Report, Federal Information Processing Standard for Federal Key Management Infrastructure, http://csrc. nist.gov/key recovery/, Nov. 1998.

Appendix: Definition of Extended Cryptographic Timed Petri Net(ECTPN)

(1) ECTPN structure

An ECTPN, which is a modeling and analysis model for protocols, is defined by modifying definitions of Petri nets [13], timed Petri net [14] and cryptographic timed Petri net (CTPN)[16] as follows:

ECTPN = (P, T, I, O, M)

- s a collection of classes of *places* such that zero or more

normal places, *data* places and *message* place.

Size of place is optionally annotated to the places in case of data and message place.

- is a collection of classes of *transitions* such that zero or more *normal*, *component* and *macro* transitions. Execution time (i.e., delay) of transitions are optionally annotated to the transitions.
- ⊂ ×T is a set of *input* functions (i.e., a set of input arcs to transitions).
- ⊂ ' ×T is a set of *output* functions (i.e., a set of output arcs from transitions).
- ⊂ ' ×NI is a set of *markings* (where, NI is a set of non-negative integers).

Note that a *token,* representing by a big dot, in data or message place is interpreted as availability of data which is annotated to the place (e.g., a private-key or message is available). A token, however, in a *normal* place is interpreted as an abstract entity of control or data as in the case of normal Petri nets.

A *component* or *macro* transition is interpreted as a function or component such as RSA, DES, exclusive-or(⊕ merge, divide, send and receive, of which execution times can be labeled. A normal-transition is an '*immediate transition'* (i.e., zero delayed firing timed transition) as in the case of normal Petri nets. A *normal* transition is interpreted as an AND logic of input places. For the purpose of readability of the net, pictorial notation of elements of an ECTPN is defined in Fig. 8.

(2) Marking and firing rule

- l places are *marked* whenever data or a message is available to the places by means of initialization and initial key distribution. Note that in some data places, which represent a result of integrity check, the data place is marked only if the result of the integrity check is true. A set of marked places is a *marking set*. A marking set is a state of marked area of system which is modeled by an ECTPN.
- l transitions whose input places are marked are *enabled*.
- ıy pairs of enabled transitions whose sets of input places are overlapped are *exclusively enabled*.
- l non-exclusively enabled transition are to be *fired* concurrently.
- ·ing transitions all output places of the transitions are marked after delay time annotated to the transition, and result in a new marking set.

(3) Properties of ECTPN

An ECTPN is not a class of extended Petri net, but *colored Petri nets*. An ECTPN is defined and used for the purpose of pictorial specification and analysis (e. g., reachability, performance) of a multiple agent based key recovery protocol. All of the formal properties of an ECTPN is inherited from CTPN [16], Time Petri net [14] and Petri net [13]. Note that an ECTPN model is an integrated model of data flow and control flow of the multiple agent based key recovery protocol.

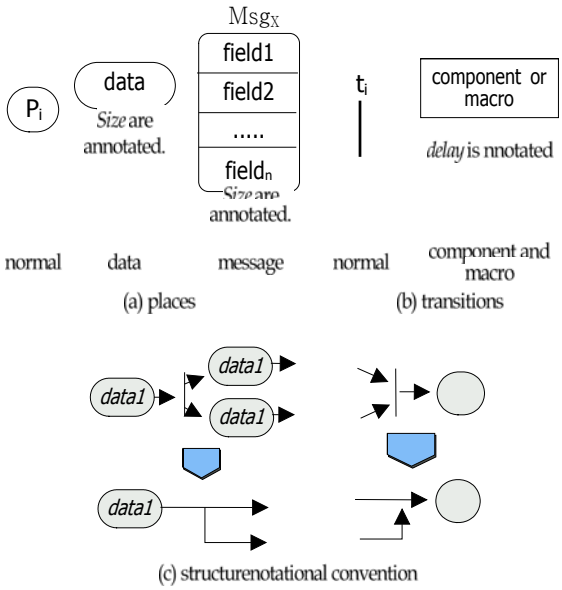


Fig. 8. Pictorial notation of elements of extended cryptographic timed Petri net