Mechanisms for Anonymous Memories

Mechanisms for Anonymous Memories

Shinsuke Tamura, Hazim Anas Haddad, Hiroya Tsurugi and Alam Kazi MD. Rokibul

Graduate School of Engineering, University of Fukui
tamura@fuis.fuis.fukui-u.ac.jp

***Abstract*-This paper discusses requirements for anonymous memories, and proposes their implementation approaches with possible applications. The anonymous memory is a set of memory sections assigned to anonymous owners of memory sections, and enables the owners to maintain their sensitive information securely without disclosing their identities even to the manager of the memory system. Possible industrial applications include the remote maintenance, in which maintenance companies maintain machines located at remote factories without knowing owners of machines.**

I. INTRODUCTION

An anonymous memory is a set of memory sections that are owned by anonymous owners. Owners can maintain their sensitive information securely while concealing their identities from others including managers of the memory. In recent business and other activities, not only information itself but also owners of the information are required not to be disclosed, and this is true also for activities in industrial sectors. For example, currently, machine maintenance companies can know not only operation histories of machines to be maintained but also names of factories where the machines are located [8], but the owner of the machines do not want maintenance companies to know their names in order to keep their production secrets confidential from their competitors. To satisfy these requirements, in the following sections, the anonymous memory is proposed. Requirements, possible implementation approaches and applications are also discussed.

II. REQUIREMENTS FOR THE ANONYMOUS MEMORY

Fig. 1 shows the configuration of the anonymous memory. It consists of multiple memory sections that are assigned to individual owners of the sections. All operations on these sections are executed by the memory manager, e.g. owners access their memory sections through the manager. The memory manager consists of registration, access control, memory access, error detection and accounting parts, and registers/deregisters owners, authenticates owners to protect the memory system from being accessed by unauthorized entities, executes read/write operations on memory sections, detects illegal memory operations, and charges owners for their memory uses, respectively. Owners of memory sections and the memory manager are connected through anonymous networks to enable owners to access their memory sections while concealing their network addresses.

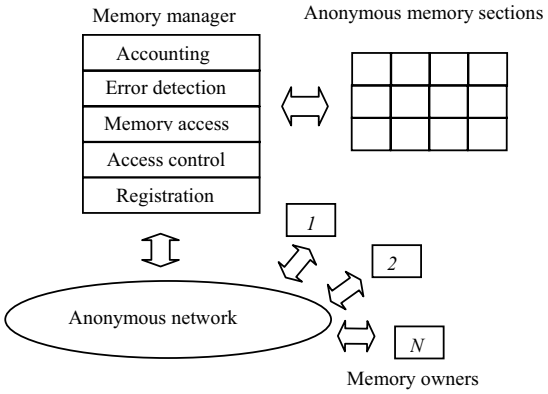

Fig.1 Configuration of the anonymous memory

To store information securely while maintaining anonymity of owners, the anonymous memory should satisfy the following requirements.

Anonymous owners: Owners must be able to read/write data from/to their memory sections without disclosing their identities. At the same time, it must be ensured that only authorized entities are allowed to access their memory sections. Also in order to achieve anonymity of owners that access their memory sections through networks, owners must be able to send their messages and receive responses while concealing their network addresses.

These requirements can be satisfied by currently available technologies for anonymous authentication and communication. However, anonymous memories must have the following additional features. Namely, different from other systems, in which users are registered and deregistered to or from the systems while disclosing their identities, in the anonymous memory system, memory section owners must be registered or deregistered to and from the system anonymously, because fixed memory sections are assigned to individual owners. When owners are registered or deregistered while showing their identities, the memory manager can easily identify linkages between memory section owners and their owning memory sections by finding memory sections that are created/discarded just after/before the registration/deregistration of owners.

Regarding to the anonymous communication, to use memory sections for general data retrievals, anonymous networks should maintain their efficiency almost at the

same level as that of non-anonymous ones. Although various kinds of Mix-nets [1], [6] successfully achieve anonymous communications, they are based on asymmetric encryption mechanisms, and therefore their communication overheads are not small enough compared with that of non-anonymous networks. The anonymous memory requires anonymous networks with much less overheads.

Secure data maintenance: Various kinds of causes about data corruptions exist for memory systems, e.g. the memory system may not accomplish operations requested by memory users, memory users may intentionally or accidentally put invalid data in their memories, malicious third parties may corrupt data, etc. To prevent data corruptions or to recover from data corruptions when preventions are difficult, any memory system must identify cause events that lead data corruptions. However, the establishment of mechanisms to identify cause events that introduce data corruptions in the anonymous memory is more difficult than in usual memory systems, because data in memory sections are read or written by anonymous owners; and mechanisms that resolve disputes about data corruptions between the memory manager and owners become important.

Also, the volume of evidences that should be maintained by memory section owners for resolving disputes must be reduced as much as possible. Because these evidences are sensitive, owners cannot keep them in their private memories when their volumes are large, i.e. owners should maintain these evidences in other secure memory systems managed by other entities. However, maintaining data in various different systems is inconvenient and also error prone for memory section owners.

Anonymous payment: The memory manager should be able to charge owners fees for their uses of memory sections at the end or the beginning of its every service period, without knowing their identities. At the same time memory section owners must be able to protect themselves from the memory manager's charges of excessive fees.

III. Possible Implementation Approaches

A. Anonymous authentication

This subsection proposes anonymous authentication mechanisms that satisfy the requirement about anonymous owners discussed in the previous section, except the ones about anonymous registration and deregistration of owners. Mechanisms that enable anonymous registration and deregistration are discussed in Sec.3.D.

The authentication mechanisms in anonymous memory systems must satisfy the following requirements, i.e.

1) Identities of memory section owners that are accessing their memory sections must not be disclosed to any entity except the owners themselves, and
2) The memory manager must accept requests for accessing particular memory sections only from owners of the corresponding memory sections.

Anonymous tokens based on blind signature [4][7] and several additional mechanisms proposed here satisfy the above requirements. A mechanism for anonymous tokens works as follows. Namely, the memory manager M authenticates memory section owner O only when O shows a token that has M's signature and it is not used repeatedly. Here, each token consists of a unique number, and M issues a new token to O for its next authentication request while blindly signing on it, in exchange for the token that O is currently using. Then, because only authorized owners have tokens with M's signature, and M signs on tokens without knowing their contents, O can prove its eligibility without disclosing its identity.

In more detail, O chooses a number T_n that is unique in the system before its (n-1)-th authentication request, and encrypts T_n into $E_O(T_n)$ by its secret encryption key K_O. Then, M signs on $E_O(T_n)$ by its secret key K_M to generate $S_M(E_O(T_n))$. Here, the result $S_M(E_O(T_n))$ is M's blind signature on T_n. Namely, O can generate a token $S_M(T_n)$ while decrypting $S_M(E_O(T_n))$ by its secret decryption key K_O^{-1}; and anyone can reconstruct T_n from $S_M(T_n)$ by using K_M^{-1}, the public key of M. However, M cannot identify O from $S_M(T_n)$, because M signs on $E_O(T_n)$ without knowing T_n.

It is trivial to link individual memory sections to their anonymous owners. Manager M can identify the linkage between a memory section and its owner O by only memorizing blinded token $E_O(T_n)$ that O showed to M at its previous ((n-1)-th) authentication request. Namely, M can decide that O has the right to access its requesting memory section when O's showing token $S_M(T_n)$ is consistent with blinded token $E_O(T_n)$ corresponded to the memory section. Additive encryption algorithms described in subsection 3-C enables M to determine whether $S_M(T_n)$, O's showing token, is consistent with blinded token $E_O(T_n)$ that is corresponded to the memory section, without knowing O's secret key.

For the purpose to conceal only the identity of owner O, O can ask M to sign on T_n without encrypting it, if O had acquired initial signed token $S_M(T_1)$ anonymously. However if T_n is not blinded, M can know the exact memory section corresponding to T_n, M can also generate copies of $S_M(T_n)$ by itself, and unnecessary disputes about impersonation will occur, e.g. O can complain that M issues its token $S_M(T_n)$ also to P while giving a copy of $S_M(T_n)$ to P. Therefore O should show its unique number T_n while encrypting it into $E_O(T_n)$. Here, O can generate number T_n that is unique in the system by picking it from the unique number-list prepared by M. Of course O should pick numbers anonymously through the anonymous authentication mechanism, for an example, through a mechanism proposed in [9], and to force O to pick numbers only from the unique number-list, M must signs on numbers in the list while using a secret key different from K_M.

Regarding to impersonation, M knows numbers that owners picked from the unique number-list; therefore M can still generate copies of valid tokens and give them to other entities. Although impersonations themselves caused by these copies are not serious, because M does not know

correspondences between these copies and their corresponding memory sections, however owners can claim that *M* maliciously disclose their tokens while intentionally disclosing their tokens to others. Disputes of this kind also can be resolved completely by implicit transaction links discussed in Sec.3.C. Namely, the manager and owners can agree about the ownerships of tokens by checking the latest memory access logs of owners while maintaining anonymity of owners.

B. Anonymous network

Among of various kinds of mechanisms that enable entities to send messages without disclosing their identities, Mix-net [1],[6] is the most known and effective mechanism. However, conventional mix-nets must adopt asymmetric encryption algorithms; therefore although they are appropriate for applications with short message exchanges such as ones in electric voting, it is difficult to use them for general applications with heavy message traffics because of their large overheads in message encryptions and decryptions. The anonymous memory requires more efficient networks.

This difficulty can be removed by a symmetric encryption based Mix-net (SEBM) proposed in [11]. Different from usual Mix-nets, SEMB adopts symmetric encryption algorithms to conceal identities of message senders, i.e. encryptions and decryptions of data are achieved by simple secret number multiplications. Therefore entities can send large amount of data with much less overheads than in usual Mix-nets. Also, although the massage encryption mechanism is simple, it is hard to break it, because different secret numbers are applied to different messages.

C. Dispute resolution

Mechanisms proposed in this subsection relate to the requirements about secure data maintenance and a part of anonymous owners discussed in the previous section. Because the memory manager executes all operations on memory sections requested by memory section owners, disputes about data corruptions can be resolved easily after the manager's acceptance of memory read/write operation requests. Namely, the manager is responsible for any data corruptions once the manager and the owner mutually agree with that the owner's requesting operation is accepted by the manager. Therefore, when the following 3 mechanisms are established, disputes between the manager and memory section owners about data corruptions can be resolved while making the anonymous memory system practical.

1) To force the memory manager to accomplish operations requested by memory section owners when the owners show their effective tokens,
2) To force memory section owners to approve completions of their requesting operations when the manager accomplishes the operations successfully, and
3) To reduce the volume of evidences that memory section owners should maintain to resolve disputes about data corruptions as much as possible.

The 1st requirement relates to situations where intentionally or accidentally the manager does not accept requests of memory section owners although the owners are showing effective tokens, or where the manager does not accomplish requested operations correctly while accepting requests from owners. The 2nd requirement relates to cases where memory section owners claim that the manager does not handle their requests without sending the requests, or claim that the manager executes operations that they do not request. About the 3rd requirement, anonymous memories are for reducing responsibilities of memory section owners for maintaining their sensitive data. Then because these evidences are also sensitive, the volume of evidences that owners should maintain to resolve disputes must be small enough to make anonymous memories practical.

These 3 requirements can be satisfied by implicit transaction links proposed in [10]. An implicit transaction link used in an anonymous credit card system is a pair of transaction IDs that are corresponded to the current and the next transactions of a cardholder as shown in Fig. 2. Here, transaction IDs in the anonymous credit card system have the same roles as tokens in the anonymous memory. Therefore in the figure, the word "token" is used instead of the word "transaction ID." An important thing is that the next tokens are encrypted by cardholders; therefore the card company cannot know linkages of transactions executed by the same cardholders, although the card company itself generates and stores implicit transaction links. The proposed mechanism exploits this implicit transaction link, i.e. the proposed mechanism records the log of the *n*-th request of memory section owner *O* with the implicit transaction link. Here, blinded next token $\underline{E}_O(T_{n+1})$ in the implicit transaction link has a different form from $E_O(T_{n+1})$, the next token that *O* shows to *M* with its current request to be signed blindly in the anonymous authentication procedure. Different from $E_O()$, $\underline{E}_O()$ satisfies the additive property, i.e. the relation $\underline{E}_O(A+B) = \underline{E}_O(A) + \underline{E}_O(B)$ is satisfied.

Current token	Blinded next token
T_n	$\underline{E}_O(T_{n+1})$

Fig. 2 Implicit transaction link

A log of the *n*-th request of memory section owner *O* is constituted as a set of items shown in Fig.3, i.e. *RN*(*n*), *ITL*(*n*), *REQ*(*n*) and *r*(*n*). In the figure, *RN*(*n*) is the number of requests that *O* asked to execute until now (i.e. *RN*(*n*) = *n*), *ITL*(*n*) and *REQ*(*n*) represent the implicit transaction link and the request digest (*REQ*(*n*) is calculated as $\underline{E}_O$(H(*requested operation*)); where H() is a hash function), and *r*(*n*) is a random number secret from owners. Then different from *M* that should maintain logs of all requests, *O* maintains only its latest accumulated log *ACC*(*n*). *ACC*(*n*) is a set of *ARN*(*n*), *AITL*(*n*), *AREQ*(*n*), and *Ar*(*n*), and they are calculated as {*RN*(*1*) + --- + *RN*(*n*)}, {*ITL*(*1*) + --- + *ITL*(*n*)}, {*REQ*(*1*) + --- + *REQ*(*n*)}, and {*r*(*1*) + --- + *r*(*n*)}, respectively. Actually, a log is encrypted to a set of linear combinations of 4 items in

Fig.3 by applying *M*'s secret coefficients. Therefore *O* cannot forge nor modify its logs consistently. On the other hand, *M* that knows the coefficients can calculate *ARN*(*n*), *AITL*(*n*), *AREQ*(*n*), and *Ar*(*n*) from the sum of encrypted logs by solving linear equations. At the same time, *M* cannot modify logs consistently either, because *ITL*(*n*) and *REQ*(*n*) are constructed by *O*'s secret key.

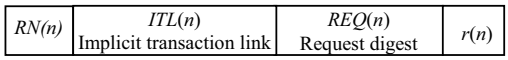

Fig. 3 Request log

Hand shaking procedure between M and O: M and *O* mutually agree with the *M*'s acceptance of *O*'s *n*-th request through the following steps, i.e.,

1) *O* requests operations on its memory section *S* while showing its *n*-th token $S_M(T_n)$, its accumulated log *ACC*(*n-1*) and the location of *S*,
2) *M* checks T_n, i.e. checks whether T_n has *M*'s signature, it is not used repeatedly, and it is consistent with $\underline{E}_O(T_n)$. Here $\underline{E}_O(T_n)$ is the next token in the implicit transaction link of the latest log about *S*,
3) When T_n passes the above checks, *M* accepts the request, and it executes requested operations,
4) When *O* confirms that its request has been accomplished, it calculates *ITL*(*n*) and *REQ*(*n*), and sends them with blinded token $E_O(T_{n+1})$ to *M*.
5) *M* calculates the log, signs on $E_O(T_{n+1})$ as $S_M(E_O(T_{n+1}))$ and returns the results to *O*.
6) *O* calculates accumulated log *ACC*(*n*) and decrypts $S_M(E_O(T_{n+1}))$ into $S_M(T_{n+1})$.

In the 2nd step, *M* can determine whether T_n is consistent with $\underline{E}_O(T_n)$ without knowing the secret key of *O*. Because $\underline{E}_O()$ is additive, when *M* asks *O* to decrypt $\underline{E}_O(t) = e_1\underline{E}_O(t_1) + e_2\underline{E}_O(t_2) + \text{---} + e_s\underline{E}_O(t_s) + e_{s+1}\underline{E}_O(T_n)$ into $t = e_1t_1 + e_2t_2 + \text{---} + e_st_s + e_{s+1}T_n$ by *O*'s secret key, *O* can calculate correct *t* even it does not know e_1, e_2, --- ,e_{s+1}, if T_n and $\underline{E}_O(T_n)$ are consistent. On the other hand, it cannot calculate *t* without knowing the random numbers e_1, e_2, --- , e_{s+1}, when T_n is not consistent with $\underline{E}_O(T_n)$. Therefore, *M* can determine that T_n is consistent with $\underline{E}_O(T_n)$ when *O* returns *t* correctly. Here, e_1. e_2, --- ,e_{s+1} are random numbers secret from *O*, and t_1, t_2, ---, t_s are test tokens, and *O* encrypts t_1, t_2, ---, t_s into $\underline{E}_O(t_1)$, $\underline{E}_O(t_2)$, --- $\underline{E}_O(t_s)$ by its secret key and sends the results to *M*, when *O* registers itself to the anonymous memory system. *M* can check the correctness of *ITL*(*n*) and *REQ*(*n*) sent from *O* in the same way.

Dispute resolution procedure; Disputes between *M* and *O* about data corruptions can be resolved as follows. Firstly, requests of *O* are ensured to be accomplished provided that *O* has effective token T_n. When the manager *M* does not accept *O*'s request or *M* executes operations not appropriately, *O* can request operations again or claim that executed operations are not the requesting ones, while showing T_n, because the latest accumulated request log *ACC*(*n-1*) enables *O* to prove that T_n is owned by *O* even after T_n and $\underline{E}_O(T_n)$ have been disclosed to *M*. Namely, when *O* calculates the sum of current tokens and that of the blinded next tokens based on *AITL*(*n-1*), the difference between $(\sum T_k - T_1)$ and decrypted $\sum\underline{E}_O(T_{k+1})$ coincides with T_n. Here, T_1 is the token that *O* used in its first access, and the correctness of *O*'s decryption can be proved in the same way as proving correctness of T_n in the 2nd step. *O* also can prove that *ACC*(*n-1*) is its latest accumulated log, because no one except *O* can comstruct *ACC*(*m*) (precisely *ITL*(*m*) and *REQ*(*m*)) for $n \leq m$ that can be decrypted into meaningful data by *O*'s secret key. Here, when *M* insists that its forging *ACC*(*m*) is *O*'s latest log, *O* should disclose its secret key in order to prove that *ACC*(*m*) is invalid. However *M* must accept the risk that it cannot continue its services when it is determined that *ACC*(*m*) is invalid. It is trivial to ensure the accomplishment of *O*'s *n*-th request after *O* has got its *n*-th request log from *M*.

On the other hand, *M* can prove that *O* has approved the accomplishment of its *n*-th request by showing T_{n+1}, token that *O* used for its (*n+1*)-th request, because *O* can get T_{n+1} only after confirming the completion of its *n*-th request. Also *O* should agree with the fact that T_{n+1} belongs to *O*, when *M* reveals its maintaining *ITL*(*n*). In the case when *O* does not return blinded token $E_O(T_{n+1})$ to *M*, the transaction of *O* does not terminate and *M* cannot continue interactions with *O*; however this does not cause any inconvenience. All entities except *O* can continue to access the memory system successfully.

About the 3rd requirement, as discussed in the above, evidences that should be maintained by individual memory section owners to prove their honesties and to ensure completions of their requesting operations are only their latest tokens and accumulated request logs.

The above mechanism does not prevent *M*'s dishonesty, in which *M* writes corrupted data in memory sections while returning consistent evidences to owners. However, these kinds of dishonesties will be eventually detected by owners when they read their memory sections, and disputes about the causes of the data corruptions are resolved by using accumulated request logs maintained by the owners as described above. A tamper resistant memory discussed in the next section can provide more efficient solutions.

D. Anonymous registration and payment

It is straightforward to apply anonymous credit card systems proposed in [10] to anonymous payment for using of memory sections. However, owners should pay for memory uses in individual operation periods before the periods start. Because owners are anonymous, the memory manager cannot identify owners that did not pay for their memory uses without complicated procedures. If payments are done before individual operation periods, even when an owner does not pay for its memory section, the memory manager can simply close the corresponding memory section.

Regarding to owner registration and deregistration, memory section owners can access their memories without

disclosing their identities after they acquire their initial tokens, even if they are registered while disclosing their identities. However, when owner *O* creates/discards its memory section immediately after it has been registered/deregistered, it is not difficult for manager *M* to link *O* and its memory section. Therefore *O* should register and deregister itself without disclosing its identity, while proving its eligibility.

This anonymous registration and deregistration can be achieved by anonymous credit card systems. Namely, the credit card system checks the eligibility for memory section owners when it checks that for cardholders at its card registration phases while identifying owners; and owners register themselves to the memory system as transactions of the credit card system to acquire their initial tokens for memory sections. Then, owners can protect them from being linked to their memory sections, because transactions of credit card systems are anonymous. Deregistration can be executed simply based on anonymous tokens that owners acquired at their last memory accesses.

However, credit card systems must not be linked to anonymous memory systems except in conjunction with registrations and payments. The reason is as follows. Namely, in anonymous memory systems, sequences of requests of the same memory section owners necessarily linked, because owners access their same memory sections. However, sequences of transactions of same cardholders must be protected from being linked, because these sequences are strong supports to identify cardholders. Therefore, tokens for using credit card systems and those for accessing memory sections must be issued as independent ones. When memory section owners use tokens of credit card systems only for registrations and payments, entities other than owners, e.g. the memory manager, can link tokens of credit card systems to those of memory systems only once a month for example. Therefore it is impossible to estimate cardholders through these linkages.

IV. USING ANONYMOUS MEMORY SYSTEMS

Memory section owners of the proposed anonymous memory must execute the following steps to acquire and use their memory sections.

1) Proving eligibility
   Memory section owner *O* registers itself to the anonymous credit card system, while showing its identity, then *O* is automatically accepted as an eligible entity for the memory system.
2) Registration
   O registers itself to the memory system without disclosing its identity by showing its current transaction ID (a token for the credit card system). Then the memory system manager *M* assigns memory sections to *O*, and O acquires its initial token (*M* registers the linkage between the encrypted initial token and the memory sections to identify the memory sections owned by *O*).
3) Accessing memory sections
   O executes operations on its memory sections while showing its current token, and acquires its next token.
4) Payment
   O pays for its memory section through the credit card system while showing its transaction ID and current token at the beginning of every service period of the memory system. As the consequence *O* acquires its next token.
5) Deregistration
   O deregisters itself from the memory system while showing its current token.

V. APPLICATIONS OF THE ANONYMOUS MEMORY

The anonymous memory has various applications, e.g. people can save their transaction records of anonymous credit card systems anonymously, so that they can recover the data even when they lose their cards. In this section, 2 possible industrial applications are discussed, the one is the remote maintenance and the other is the tamper resistant memory.

A. Remote maintenance

Fig.4 shows an example of the configuration of the anonymous memory for remote maintenance. Owners of machines put operation states of their machines in their anonymous memory sections, and maintenance companies maintain machines by the data stored in the memory sections without knowing the machine owners. There are at least 2 advantages to use the anonymous memory, i.e. not to connect machines and maintenance companies directly through anonymous networks. Firstly, the volume of messages put into anonymous communication channels can be distributed over time. In cases where machines and maintenance companies are connected directly through anonymous networks, the volume of messages increases as maintenance companies start their maintenance jobs. On the other hand when anonymous memories are used, machines can send their state information to their corresponding anonymous memory sections through anonymous networks that are not efficient as non-anonymous ones bit by bit, not as batch data. Secondary maintenance data of multiple machines can be efficiently distributed and merged so that they are seen as different virtual machines to protect owners of machines from being identified through histories of their states as discussed below.

A serious problem of remote maintenance systems discussed here is the fact that in many cases operation histories of machines located at a factory provide good suggestions to estimate the factory. Maintenance companies may estimate machine owners when the owners provide the maintenance companies with complete histories of operation states of their machines. Theoretically, there are methods to calculate any kind of statistics from encrypted data [2][3][5], i.e. without disclosing exact raw data, however they are not practical, because they require a lot of computations and communications. Therefore this subsection discusses the other 2 possibilities although they are strongly problem dependent. In the figure, memory sections are divided into *2* levels, i.e. level-*1*, in which raw operation data of machines

are stored, and level-*2*, from which maintenance companies acquire data necessary for maintenances. Here, selected data in the level-*1* memory sections are transferred to the level-*2* memory sections, so that memory sections in the level-*2* constitute virtual machine groups consists of combinations of multiple machines located at different places (sometimes owned by different owners). Therefore it becomes difficult for maintenance companies to identify real owners of machines from operation histories of virtual machine groups they are maintaining. Another solution is to exploit homomorphic encryption algorithms. Here, homomorphic encryption function E() satisfies E(*A*) + E(*B*) = E(*A*+*B*), E(*A*)E(*B*) = E(*AB*), or E(*A*)E(*B*)= E(*A*+*B*). Therefore, when machine owners encrypt operation histories of their machines based on these encryption algorithms to be put in the anonymous memory sections, it is possible to divide maintenance functions into modules, and distribute these modules to maintenance companies and machine owners, so that the maintenance companies calculate averages, variances, auto/mutual correlations, etc. of encrypted operation histories, and the machine owners decrypt them to send the results back to the maintenance companies.

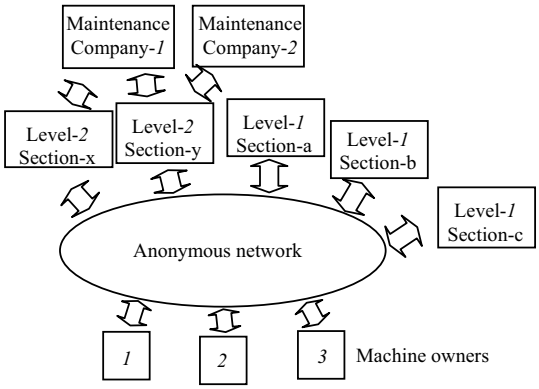

Fig.4 Remote maintenance

B. Tamper resistant memory

A tamper resistant memory section is a memory section that can protect the data in it from being modified by unauthorized entities. One form of a tamper resistant memory section is a duplicated one, i.e. a memory section that has multiple copies. Data owners can check correctness of their data by comparing multiple copies; they can detect illegal data modifications when data in multiple copies are not consistent, and they can recover the modified data along the majority decision principle.

A proposed tamper resistant memory based on the anonymous memory shown in Fig. 5 enhances the security of these duplicated memory sections. Namely, different from normal duplicated memory sections, in which entities, e.g. system managers, can illegally modify data without being detected by modifying data in all copies, data in anonymous memory based duplicated memory sections are difficult to modify, because unauthorized entities that try to modify the data do not know locations of the copies.

To make the tamper resistant memory more secure, asynchronous and fake memory access mechanisms must be implemented, because entities can identify locations of anonymous copies by eavesdropping simultaneous memory access messages. Also the data in different copies must be encrypted by different keys, in order to disable eavesdroppers to identify locations of anonymous copies by comparing their contents.

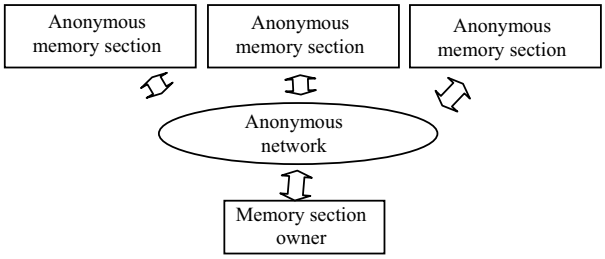

Fig. 5 Tamper resistant memory

VI. CONCLUSION

This paper discusses requirements for the anonymous memory. Approaches to satisfying the requirements and 2 industrial applications, i.e. remote maintenance and tamper resistant memories are also discussed. Although it is not discussed, SAAS (Software As A Service) is one of the possible important applications.

REFERENCES

[1] D. Chaum, "Untraceable electronic mail, return address and digital pseudonyms," *Communications of the ACM*, Vol.24, No.2, pp. 84-88, 1981.

[2] A. C. Yao, "How to generate and exchange secrets," *Proc. of the 27th IEEE Symp. on Foundations of Computer Science*, pp.162-167, 1986.

[3] O. Goldreich, M. Micali and A. Wigderson, " How to play any mental game," *Proc. 19th ACM Symp. on Theory of Computing*, pp.218-229, 1987.

[4] S. Brands, "Untraceable off-line cash in wallets with observers," *Proc. Of CRYPTO'93*, pp.302-318, 1994.

[5] M. Naor, B. Pinkas and R. Sumner, "Privacy preserving auctions and mechanism design," *Proc. of the 1st ACM Conference on Electronic Commerce*, pp.129-139, 1999.

[6] B. LeeB. B. Dawson, K. Kim, J. Yang and S. Yoo, "Providing receipt-freeness in Mixnet-based voting protocols," *Information Security and Cryptography ICISC2003 6th International Conference*, 2003.

[7] R. Shigetomi, A. Otsuka and H. Imai : "Refreshable Tokens and Its Applications to Anonymous Loans," *SCIS2003*, 2003.

[8] D. Djurdjanovic, J. Lee and J. Ni, "Watchdog agent - An infotronics based prognostics approach for product performance assessment and prediction", *International Journal of Advanced Engineering Informatics*, Special Issue on Intelligent Maintenance Systems, Vol.17, No.3-4, pp.109-125, 2003.

[9] S.Tamura and T. Yanase, "Information sharing among untrustworthy entities," *IEEJ Trans*. EIS, Vol.125, No.11, pp.1767-1772, 2005.

[10] S.Tamura and T. Yanase, "A mechanism for anonymous credit card systems," *IEEJ Trans*. EIS, Vol.127, No.1, pp.81-87, 2007.

[11] S. Tamura, K. Kouro, S. Sasatani, K. MD. R. Alam and H. A. haddad, "An information system platform for anonymous product recycling," *Journal of Software*, Vol.3, No.6, pp.46-56, 2008.