Improvements on Sun et al.'s Conditional Access System in Pay-TV Broadcasting Systems

Jung-Yoon Kim and Hyoung-Kee Choi

Abstract

A Conditional Access System (CAS) proposed by Sun *et al*. has a critical security weakness in its inability to preserve backward secrecy; a former subscriber can still access programs despite his or her change in status. This weakness in Sun *et al*.'s CAS originates because 1) no change is made to a group key after a new member arrives, and 2) updates of group keys are done in an insecure manner. We show how simple protocol changes can fix these weaknesses and thus render Sun *et al*.'s CAS capable of preserving backward secrecy.

I. Introduction

A CAS is a security system designed to ensure that only authorized subscribers can access broadcasting services [1]–[4]. The existing CASs in pay-TV broadcasting systems can be classified into three models: pay-per-channel (PPC), pay-per-view (PPV), and flexible pay-per-channel (F-PPC). In PPC, a subscriber leases subscription packages among multiple groups of channels for a fixed period, typically for a month or a year. A subscriber can watch all of the programs broadcast on the channels of the groups in his subscription. Members in PPC, however, are unable to subscribe an arbitrary combination of channels according to his preference. In contrast, PPV provides a fair service, because a PPV subscriber can pay for one program at a time. However, PPV makes the subscriber inconvenient because of the high subscription frequency and low flexibility of channel selection. F-PPC further improves PPV and PPC by accommodating efficient membership management, flexible channel selection, and fairness [4].

To prevent unauthorized access in pay-TV broadcasting systems, scramble and encryption algorithms are commonly used for secure media delivery and channel protection. The encryption keys

must be distributed to all subscribers so that they can receive and decrypt the broadcasts they are entitled to under the terms of their subscriptions.

In addition to ensuring secure distribution of the group keys, backward and forward secrecy are essential security requirements in group-key management. The backward secrecy implies that new group keys must be inaccessible by former group members; the forward secrecy implies that previously used group keys must be inaccessible by new group members [5].

Huang *et al*. [2], Liu *et al*. [3], and Jiang *et al*. [6] proposed four-level key hierarchy CASs for PPV and PPC. Their approaches aim at efficient group-key distribution in terms of the number of messages sent and computational overhead requirements. Jiang *et al*.'s CAS requires G messages to deliver a group key, where G is the number of groups. Liu *et al*.'s and Huang *et al*.'s CASs further reduce the number of messages to one. Although most CASs perform a modular exponentiation as a way to handle a group key, Huang *et al*.'s CAS employs lightweight operations such as XOR, hash, and symmetric encryption. Wang *et al*.'s CAS has an advantage over other approaches in its support of diverse billing strategies by service providers [7].

Sun *et al*. proposed a new CAS for F-PPC [4]. This new CAS is more efficient and flexible than the classic CASs because of its efficiencies in transmission and storage. Despite Sun *et al*.'s CAS advances, however, we have found critical weaknesses in their management of group keys; in fact, these flaws leave the backward secrecy vulnerable to exploitation by former subscribers. In turn, pay-TV broadcasting systems are exposed to serious threats of content piracy and resultant financial loss. The critical flaws in Sun *et al*.'s CAS originate because 1) the group key remains unchanged even if a new member joins a group, and 2) group keys are updated in an insecure manner when a member leaves a group. In this paper, we analyze the weaknesses in Sun *et al*.'s CAS and then propose an improvement that eliminates these flaws and ensures the backward secrecy.

II. Review of Sun *et al*.'s CAS

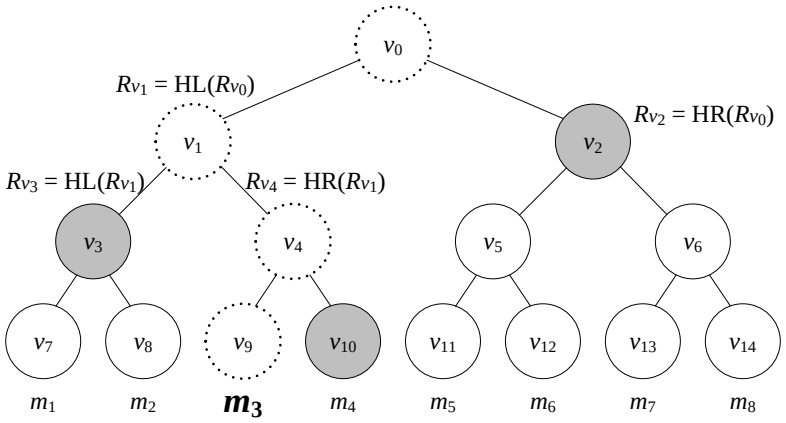

Fig. 1 An illustration of a binary tree used to manage RGK. I_{m_3} for m_3 located at v_9 should include all of the secret values in the tree except R_{v_0}, R_{v_1}, R_{v_4}, and R_{v_9}. As a result, I_{m_3} contains R_{v_2}, R_{v_3}, and $R_{v_{10}}$. The rest of the secret values are derived.

In general, key management in a CAS consists of a four-level key hierarchy. Each channel is encrypted with a control word (CW). A server refreshes the CW at a specific interval that can range from 5 to 20 seconds. The CW is distributed to legitimate subscribers after being encrypted by a channel key called an authorization key (AK). For the daily or weekly refreshment of the AK, a receiving group key (RGK) is assigned to each subscribed group and is used to encrypt the AK. Consider a system with N channels and S subscribers. Without the group key, the server would have to send N messages for all channels to update the CW and $N \times S$ messages for all members every day or week to update the AK. However, this process has been made more efficient with the introduction of the RGK because the server now sends the same number of messages to update the CW but N messages for all members to update the AK. A master private key (MPK) is a pre-shared key between a client and the server and is used to encrypt the RGK for secure distribution of group keys. The group key is updated whenever a change occurs in group membership.

Sun *et al*. proposed a new CAS, also based on a four-level key hierarchy, for F-PPC broadcasting systems. To manage an RGK, the server and group members maintain the structure of a binary tree for a group, as shown in Fig. 1. Every node (v_k) in the tree has its secret value, R_{v_k}. Every member (m_i) is assigned to a leaf node and given an I_{m_i} by the server through a secure channel. The I_{m_i} is a set of secret values of m_i, and contains all the secret values in the tree except the restricted secret value. The restricted

secret value of m_i is referred to as a secret value of the leaf node to which m_i is assigned. As illustrated in Fig. 1, I_{m_3} for m_3 located at v_9 should include all the secret values in the tree except R_{v_9}.

In order to save storage space, the size of I_{m_3} is reduced to contain only R_{v_2}, R_{v_3}, and $R_{v_{10}}$ (nodes with the shaded background in Fig. 1). The remaining secret values can be derived using two hash functions, HL(·) and HR(·), for left and right children, respectively. The restricted secret value also extends to include those secret values on the path from its node to the root. R_{v_0}, R_{v_1}, R_{v_4}, and R_{v_9} are the restricted secret values (nodes with dotted lines in Fig. 1) for m_3.

When a member m_k leaves a group G_j, all the members in the group update a group key RGK_j to RGK_j' according to

$$RGK_j' = RGK_j \oplus R_{G_j,m_k} \quad (1)$$

where R_{G_j,m_k} is a secret value corresponding to an m_k's leaf node for G_j. The server notifies members of the m_k's departure by broadcasting the identity of m_k. The departing member m_k cannot update the group key because R_{G_j,m_k} is the restricted secret value of m_k. In case m_3 leaves the group, the rest of the members update the group key by XORing the current group key with R_{v_9}. However, m_3 cannot update the group key because it cannot derive R_{v_9} from I_{m_3}. Sun *et al*. argued that this inability to update the group key would guarantee backward secrecy.

When a new member m_{n+1} joins the group, m_{n+1} receives a package of information from the server, including the current group key RGK_j and $I_{m_{n+1}}$, after encrypting this information with MPK_{n+1}. The server broadcasts an identity of m_{n+1} to all the group members in the arrival message. It would be preferable to locate a rejoining member m_3 to the node v_9 where m_3 was originally assigned. However, if v_9 is occupied by another member m_9, the server and group members append two children nodes to v_9, and then move m_9 to the left child and assign m_3 to the right child, respectively. The corresponding I_{m_3} and I_{m_9} values extend to reflect these changes in the binary tree. It is critical that all of the group

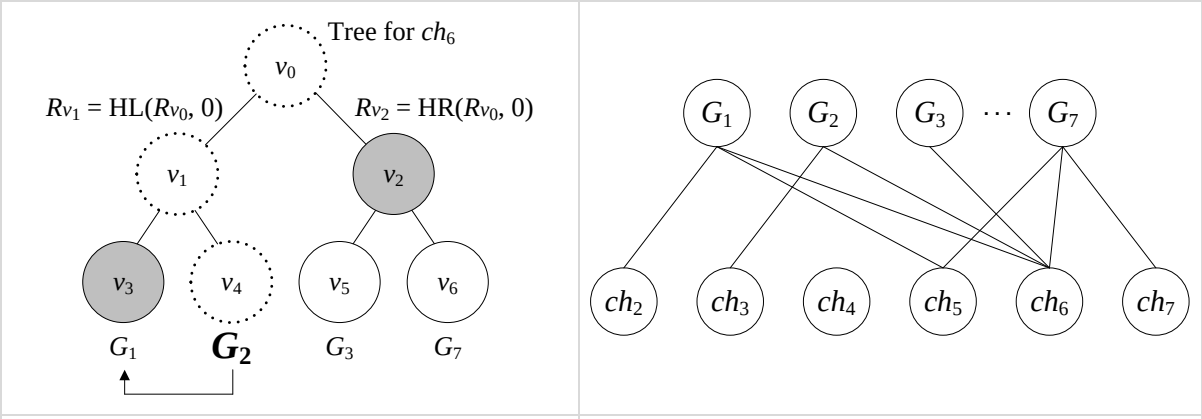

Fig. 2 (a) Illustration of a binary tree for the common channel, ch_6, when a member m_k changes the group from G_2 to G_1. R_{v_3}, the restricted secret value of G_1, should be changed because m_k knows R_{v_3}. (b) ch_2, ch_5, and ch_6 are belong to G_1. ch_3 and ch_6 are belong to G_2. ch_6 is a common channel of G_1, G_2, G_3, and G_7.

members know the position of m_{n+1} so as to include the new member in the binary tree. Note that the members update the group key only in the departure procedure, not in the arrival procedure.

The management of AK is similar to that of RGK except that an AK controls access to a channel. A binary tree for a channel ch_c is used to manage subscription packages that subscribe to ch_c and takes a group as a node of the tree as shown in Fig. 2 (a).

Note that the departure message is sent in plaintext. If m_k leaves a group G_j, RGK_j needs to be updated for backward secrecy, and all the authorization keys of the channels in G_j should also be updated. If a channel ch_c in G_j is also a channel in G_k, all the members of G_j and G_k should update the corresponding authorization key (AK_c) of the channel ch_c. In this case, if the departure messages are encrypted, the server should broadcast the departure message two times, each one encrypted with RGK_j and RGK_k, respectively. In a worst case scenario, the server may have to broadcast the departure message G times, where G is the number of groups, after encrypting the message with each group key for all the groups. The comparison in Table IV [4] shows that the number of transmitted messages for un-subscription is constant. This figure confirms that the departure message is broadcast once in plaintext to all the members. The departure message could be encrypted with the group keys at the expense of additional computational and communication overheads.

The arrival message that is broadcast is not encrypted either. Consider the example shown in Fig. 2 (b) in which a member m_k in a group G_2 subscribes to channels ch_3 and ch_6; in particular, ch_6 is shared with groups G_1, G_3, and G_7; m_k changes its subscription package to the group G_1, as shown in Fig. 2 (a). In this case, G_1's restricted secret value in the tree for ch_6 needs to be changed because m_k knows this value. According to [4], the server updates R_{v_3} to $R'_{v_3} = \mathrm{HL}(R_{v_1}, 1)$ and sends R'_{v_3} to members in G_2. Members in G_3 and G_7 are able to calculate R'_{v_3} by the fact that m_k has left G_2 and has joined G_1. This means that each member should be able to track the arrivals and departures of members of other groups. Hence, if the arrival messages are encrypted, in the worst case, the arrival messages are sent G times after encryption with each group key for all the groups. However, the authors of [4] do not consider this additional overhead. This confirms that the arrival messages are broadcast once in plaintext to all the members.

III. Cryptanalysis of Sun *et al.*'s CAS

Sun *et al*. claimed that their CAS could guarantee backward secrecy because 1) the server updates the group key using the departing member's restricted secret value, and 2) a member cannot have the restricted secret value. However, we call attention to the fact that to the contrary, Sun *et al*.'s CAS has critical weaknesses in preserving backward secrecy; a member can acquire its restricted secret value fairly easily even if the arrival and departure messages are encrypted.

When a malicious member m_3, in Fig. 1, leaves a group G_j, a group key RGK_j is replaced with $RGK_j{}'$ according to

$$RGK_j{}' = RGK_j \oplus R_{G_j,m_3} = RGK_j \oplus R_{v_9}. \quad (2)$$

Before leaving G_j, m_3 takes a snapshot of the binary tree for the group and saves the snapshot with the group key. As soon as the membership is canceled, the malicious member m_3 tries to join the group again immediately. In the meantime, the group key may be updated to another group key RGK_j'', because other

members may leave the group. When m_3 rejoins the group, the server transmits to m_3 both RGK_j'' and I_{m_3} encrypted with MPK_3, and assigns m_3 to its originally assigned node v_9 or to its right child if v_9 is already occupied (e.g., v_{20}).

If no members have left the group since m_3's departure, m_3 can then compute its restricted secret value R_{v_9} (or $R_{v_{20}}$) by using

$$R_{v_9} = R_{G_j,m_3} = RGK_j' \oplus RGK_j. \quad (3)$$

Even if some members have left the group since m_3's departure, backward secrecy is nevertheless at risk. By comparing the old and new snapshots, m_3 can find the change of membership status for other members and positions of the departing and arriving members in the group. Note that m_3 cares only for the departing members because joining does not change the group key. By knowing the positions, m_3 can further derive restricted secret values of those members in the old snapshot. Because the current group key (RGK_j'') is computed by XORing the old group key (RGK_j) with the m_3's old restricted secret value and restricted secret values of departing members, a calculation of m_3's old restricted secret value is quite straightforward. The m_3's current restricted secret value is the same as the old restricted secret value if m_3's position does not change or is the hash of the old restricted secret value if m_3's position changes.

The following example elaborates how m_3 possibly finds those positions. A malicious attacker m_3 is a member of the group G_2 with a restricted secret value of R_{G_2,m_3}. As a member, m_3 has a group key, RGK_2. This malicious member initiates an attack by leaving the group G_2. m_3 saves the group key and a snapshot of G_2's binary tree. G_2's group key is updated to RGK_2' after m_3's departure as follows,

$$RGK_2' = RGK_2 \oplus R_{G_2,m_3}. \quad (4)$$

As soon as the membership is canceled, m_3 joins the group G_2 again immediately. According to the joining protocol of Sun *et al.*'s CAS, The server must place a returning member either its last position or the right child of the last position if the last position is occupied. In the meantime, the current group key is RGK_2'', because other members may leave the group. Again, as a member, m_3 receives the group key,

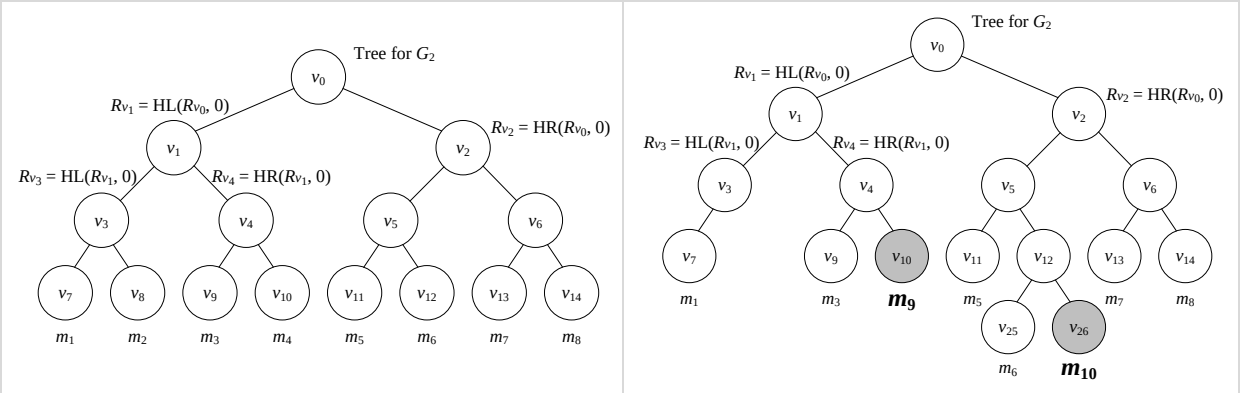

Fig. 3 Attacker's two snapshots of the binary tree. In m_3's nonmember period, m_2 and m_4 have left the group, and m_9 and m_{10} have joined the group; (a) an old snapshot of the binary tree for the group G_2 and (b) a new snapshot of the binary tree for the group G_2.

RGK_2'', and the binary tree of G_2. m_3 has a new snapshot of G_2's binary tree. The main stage of the attack is to identify which members have left the group between the time of m_3's departure and return and further to find positions of those departing members. m_3 cares only for the departing members, because joining does not change the group key. Identifying a departing member is quite straight forward in case the departure message is not encrypted. Even if the departure message is encrypted, the identification is still possible by comparing the old and current snapshots. Furthermore, this comparison makes it possible to find the departing members' positions in the old snapshot.

Fig. 3 (a) and (b), respectively, show the old and current snapshots of the group G_2's binary tree from the perspective of the attacker m_3. In m_3's nonmember period[1], m_2 and m_4 have left the group and m_9 and m_{10} have joined the group. This fact is straightforward to m_3 by comparing the two snapshots; that is, in the current snapshot, the node v_8, which was a leaf node occupied by m_2, is disappeared, and the node v_{10} is now associated with m_9 instead of m_4; furthermore, m_6 is relocated to the node v_{25} by the arrival of m_{10}.

By knowing the positions, m_3 can compute restricted secret values of those departing members. Furthermore, m_3 has RGK_2'' and RGK_2. Because the current group key (RGK_2'') is computed by XORing the old group key (RGK_2) with the m_3's old restricted secret value (R_{G_2,m_3}) and restricted secret values of departing members, a calculation of R_{G_2,m_3} is quite straightforward. Because m_3's position in the binary

[1] A "nonmember period" is the time it takes for a member to leave the group and then join this group again.

tree is either the last position or the right child of the last position, m_3 can derive the current restricted secret value readily. As a consequence, even if m_3 leaves the group again, future group keys remain available to m_3.

This attack should fail in case there is a member who had joined the group later than m_3's departure and left the group before m_3's rejoining; in other words, the attack cannot succeed if a new member joins and leaves the group within m_3's nonmember period. When this member leaves the group, the group key is updated with its restricted secret value. m_3 is unable to find this value, because the leaving member's position is not recorded in the two snapshots.

Nevertheless, we believe that the attack is still plausible because it is planned ahead of time and is manipulated purposely and automatically. Since m_3 would try to rejoin the group as early as possible, m_3's nonmember period is minimal. On the other hand, most of the regular members decide to unsubscribe channels once they find them uninteresting. The residence time of a regular member in a channel is generally longer than the nonmember period. Furthermore, a member should press a button on a remote control or a set-top box or call the service provider if he/she wants to change his/her membership. This in-hand manipulation takes longer than an automated computer program generated by a malicious attacker. Hence, it is almost impossible for a member to join and then leave the group within the nonmember period. The number of arriving or departing members in m_3's nonmember period does not affect the result of the attack as long as their status remains the same until m_3's return.

Note that Sun *et al.*'s CAS does not also achieve forward secrecy within a short haul, because a group key is not updated when a new member's arrival. When a malicious member m_3 has joined a group, m_3 can illegally retrieve any content that had been serviced between the last member departure and m_3's arrival. However, this might not be a critical problem in Sun *et al.*'s CAS, because this vulnerable period might be very short in pay-TV systems with a large number of subscribers who can change their membership arbitrarily. Nevertheless, Sun *et al.*'s CAS still needs to be improved, because its inability to achieve backward secrecy should be a serious security flaw.

IV. Proposed improvement

In light of these weaknesses, we have improved Sun *et al*.'s CAS to satisfy both backward and forward secrecy. The proposed protocol does not depart from Sun *et al*.'s CAS in any radical way. The departure consists of enhancing security of the group-key distribution so that 1) a group key is updated both when a member joins and leaves a group, and 2) the hash operation is applied to a group-key update when a member departure. The group key is computed according to (5) and (6), respectively, when a member m_k leaves and joins a group G_j;

$$RGK_j' = \mathrm{H}(RGK_j \oplus R_{G_j,m_k}), \tag{5}$$

$$RGK_j' = RGK_j \oplus R_{G_j,m_k}, \tag{6}$$

where $\mathrm{H}(\cdot)$ is a one-way hash function. Note that (6) is the same as (1).

The security weakness in Sun *et al*.'s CAS is largely a consequence of the insecure manner in which the group key is updated. The XOR operation in this update is insecure because of its reversibility in the sense that the restricted secret value can be computed by knowing the two subsequent group keys. Because the server and members do not update the group key when a new member joins, a malicious member can rather easily collect the two subsequent group keys. The proposed protocol becomes more secure by updating the group key on all arrivals and departures. It would be all but impossible for a malicious member, even after rejoining the group, to collect the two subsequent group keys.

If a group key were to be updated using only the XOR operation, backward secrecy might not be preserved in a certain situation. If a new member joined a group just right after a member m_k departure, and this new member was assigned to the same node as m_k was located in the tree, the group key would be reverted to the previous group key known to an ex-member m_k. The one-way hash operation comes in a place to resist to such group-key reversion. Note that the hash operation is unnecessary in the group-key update after new member arrivals. This is true that only a single hash operation is sufficient enough to make the group key not able to be reverted to the previously known group key.

A simple change in the proposed protocol protects the restricted secret value from disclosure. As long as a member is unable to access its restricted secret value, the protocol guarantees both backward and forward secrecy. To update a group key and the channel keys in the group when a new member joins a group, the proposed protocol introduces the additional overhead of an XOR operation per key for each member (see (6)). When a member leaves a group, the additional overhead is a simple hash operation per key for each member (see (5)). Let K stand for the number of group keys and channel keys that must be updated when a member joins or leaves a group. Overall, Sun *et al.*'s CAS requires K XOR operations and no operations, respectively, when a member departs and arrives. In contrast, the proposed protocol requires K XOR and K hash operations when a member departs and K XOR operations when a member arrives.

V. Conclusion

Sun *et al*. has proposed an efficient and flexible CAS for group-key management in pay-TV systems. Because their design goal is to reduce the overhead associated with transmission and storage, Sun *et al*.'s CAS 1) does not update a group key when a new member's arrival and 2) updates a group key using only the XOR operation when a member's departure. These two features forfeit preservation of backward secrecy. Our proposed protocol requires minimum additional overhead to repair these weaknesses by securely protecting the restricted secret value from disclosure.

VI. References

[1] B. M. Macq and J. J. Quisquater, "Cryptology for Digital TV Broadcasting," *Proc. IEEE*, vol. 83. no. 6, pp.944-957, Jun. 1995.

[2] Y. L. Huang, S. Shieh, F. S. Ho, and J. C. Wang, "Efficient Key Distribution Schemes for Secure Media Delivery in Pay-TV Systems," *IEEE Trans. Multimedia*, vol. 6, no. 5, pp. 760-769, Oct. 2004.

[3] B. Liu, W. Zhang, and T. Jiang, "A Scalable Key Distribution Scheme for Conditional Access System in Digital Pay-TV System," *IEEE Trans. Consum. Electron.*, vol. 50, no. 2, pp. 632-637, May 2004.

[4] H. M. Sun, C. M. Chen, and C. Z. Shieh, “Flexible-Pay-Per-Channel: A New Model for Content Access Control in Pay-TV Broadcasting Systems,” *IEEE Trans. Multimedia*, vol. 10, no. 6, pp. 1109-1120, Oct. 2008.

[5] P. Sakarindr and N. Ansari, “Security Services in Group Communications over Wireless Infrastructure, Mobile Ad Hoc, and Wireless Sensor Networks,” *IEEE Wireless Communications*, vol. 14, no. 5, pp. 8-20, Oct. 2007.

[6] T. Jiang, S. Zheng, and B. Liu, "Key Distribution Based on Hierarchical Access Control for Conditional Access System in DTV Broadcast," *IEEE Trans. Consum. Electron.*, vol. 50, no. 1, pp. 225-230, Feb. 2004.

[7] S. Y. Wang and C. S. Laih, "Efficient Key Distribution for Access Control in Pay-TV Systems," *IEEE Trans. Multimedia*, vol. 10, no. 3, pp. 480-492, Apr. 2008.