XML Encryption and Access Control
A comparison and a 2nd encryption model
Draft 29.10.2000

Christian Geuer-Pollmann, University of Siegen
please send comments to geuer-pollmann@nue.et-inf.uni-siegen.de

1 Introduction

In the last weeks and month, we have seen different proposals and collections of requirements for XML encryption and some papers which describe access control mechanisms for XML documents. This paper is a collection of some ideas, how both fields could fit together

1.1 Motivation

XML is highly structured data — Data Encryption is destroying internal structure of data — These two contradictions have to be brought together, without conflicting too much.

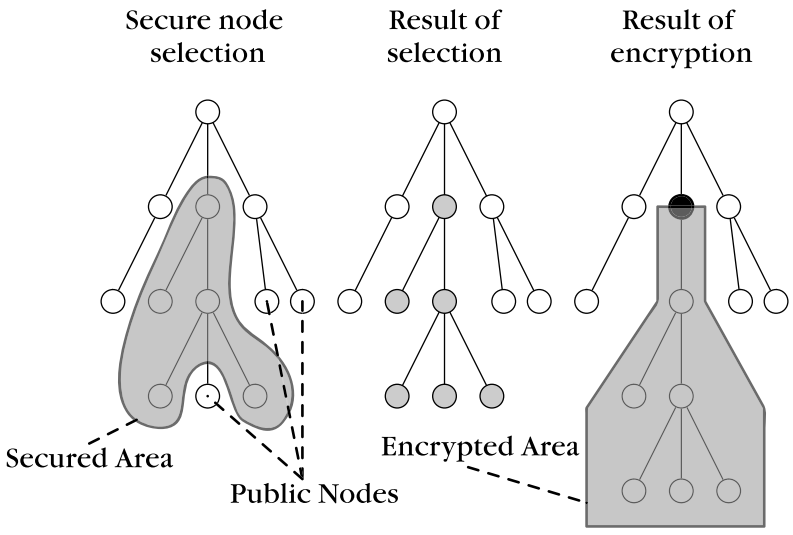
1.2 Subtree encryption (element wise)

The two published proposals by [Imamura] and [Simon, LaMacchia] have in common that they take a complete sub tree (descendant-or-self(), maybe with of without attributes of self()), serialize this subtree into a text representation, encrypt it using some encryption mechanism like a symmetric cipher and replace the unencrypted part of the document with the resulting cipher text.

The subtree encryption is an end-to-end-security approach, in which the document includes all sensitive information in encrypted (secured) form. It allows to include multiple encrypted subtrees, and depending on the choosen model and granularity, it is possible to select even single attributes for encryption.

In the following illustration, the „Public Nodes“ do not need to be confidential (encrypted), but the one at the bottom is encrypted in the subtree.

To encrypt a subtree, the nodes that should be secured are selected:

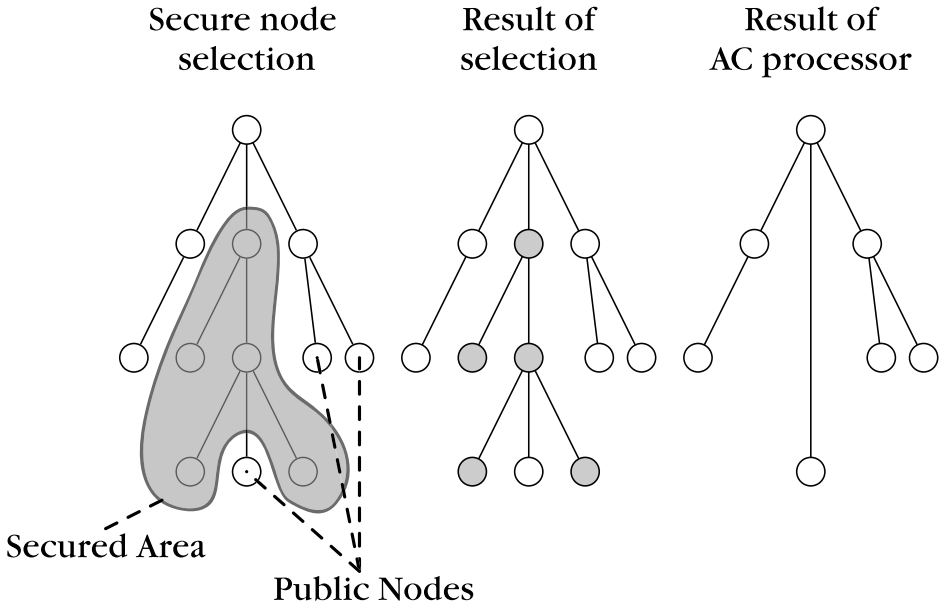


1.3 Server-side Access Control

The server-side access control scenarios by [Damiani, et.al.], [Bertino, et.al.] and [Kudo, Hada] can react more flexible in their content model:

- The XML entity is stored on a server. It is parsed into a DOM tree.
- If a client requests the entity, he is authenticated and it's authorisations are checked.
- Based on these authorisations (access control lists), the DOM tree is labeled.
- After labeling the nodes, a pruning transformation is applied; all nodes to which the client has no access permission are deleted from the tree.
  NOTE: Not only end-nodes of the tree can be deleted, even node which have childs. In this case, the childs become childs of their „grandparent".
- The last step is the serialisation of the resulting tree.

Server-side AC can completely restructure and rebuild the tree, based on the access control lists. It is not forced to make a complete subtree opaque, but it can let some elements childs visible(unencrypted) to the client without enforcing the root of the subtree (self()) being visible.



1.4 Comparison

Subtree encryption

Subtree encryption (element wise) is a good and straight-forward solution for XML Encryption and it will fit into most situations. The encrypted entity can be transferred to the client without a need for an additional encryption on the transport layer (like SSL). The XML entities can be stored encrypted on the (potentially insecure and vulnerable) web server. The decisions about access rights to different portions of the document can be made by the document creator and be immediately applied to the XML document.

Encryption has to be applied to each document individually, but in analogy to *extensible stylesheet transformations* (XLST), it should be possible to apply an „*encryption policy stylesheet*" to a XML document which allows an automatic encryption process based on a defined policy.

SERVER-SIDE ACCESS CONTROL

In contrast to this model, server-side access control has much more flexibilty in the resulting document, because the confidentiality transformation is not constrained to complete subtrees. The pruning of sensitive or classified information prevents the requesting client from accessing this information, but during the transfer to the client, there is a need for an additional encryption on transport layer (like SSL). The access control processor needs to be secure and trustworthy, because this centralised element has access to the complete information base. A disadvantage is the need to make AC decisions online.

The access rights for a specific document have to be added to the ACL (access control list) database. An advantage of this model is the ability of applying a specific ACL to a large class of documents (based on DTD/Schema).

Model	Subtree encryption	Server-side access control
Secure Complete subtrees	possible	possible
Secure only Attribute values	possible (depends on proposal)	possible
Leave „deep“ descendants visible	no	yes
Needs trustworthy server	no	yes
Automation of access/encryption decisions	needs some sort of style sheets	yes
Needs additional transport security	no (even complete instance can be encrypted)	yes
New recipients can be added without reencrypting the contents	no	only new XACL

Table 1: Comparison between the existing models (disadvantages are marked grey)

It could be nice to get the best from subtree encryption and server-side AC:

- allow unencrypted (visible) content within an encrypted subtree
- does not need a trustworthy online access control processor (only encryption, no online transformations)
- no need for additional encryption

2 The idea

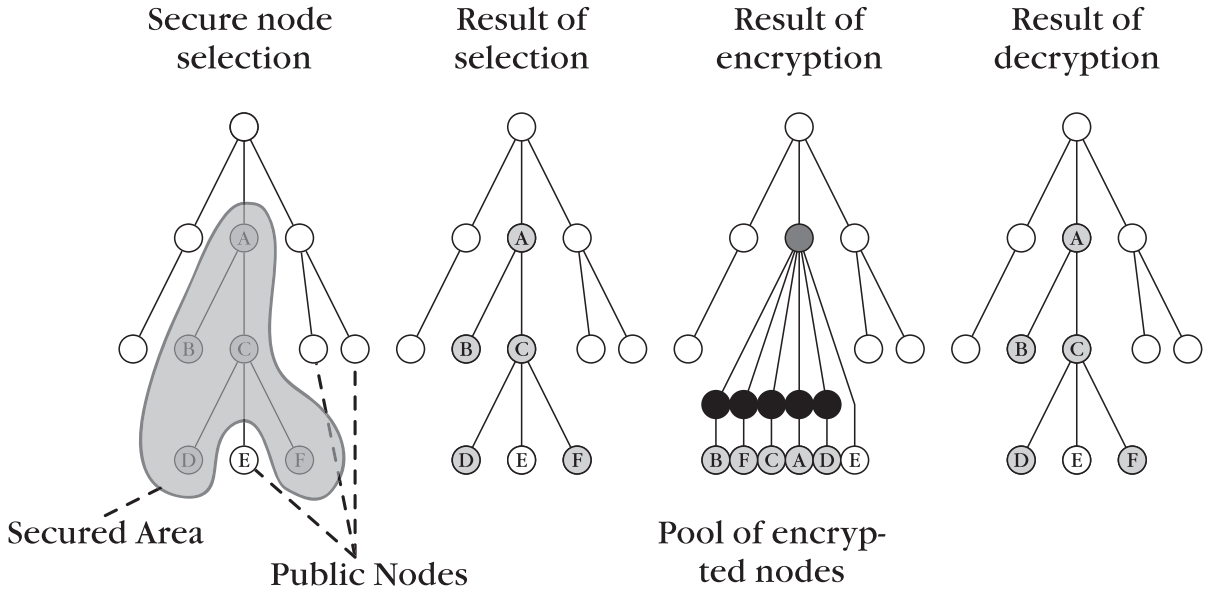
2.1 Outline

The rough idea is to take the nodes which have to be secured and encrypt each node individually(with a node-specific encryption key). Include all encrypted nodes, together with the key information, in a new element.

- Transform the XML document to a DOM tree,
- label the nodes and store the positions of each node (positions are: "What are the ance-

stor()s, descendant()s, preceding-sibling()s and following-sibling()s")

- Attach the original position information of the nodes to the nodes and encrypt each node/position with an individual key (flattening the tree and encrypting the single nodes)
- Based on the Access-Control-Lists or encryption decisions, make a package for each client, which contains the keys for the nodes which the client is permitted to see and encrypt the package for the specific client. Each client gets it's own collection of node keys, which is encrypted with a key-encryption-key.
- To decrypt the document,
  - a client picks his key package,
  - decrypts his keys,
  - decrypts all elements for which he has keys and
  - rebuilds his „personal" tree based on the position information attached to the nodes.



In the above picture, the white node is public accessible. The key information structures are not shown in the picture.

The key packages allow a level of indirection in the key management process. It is possible to group complete classes of element keys into packages, which are encrypted using a symmetric package-key. This key will be encrypted with the public key of each recipient.

2.2 Comparison

This „new" model has some advantages, as well as some disadvantages:

- The model allows the flexibility of server-side access control in conjunction with a real encryption; the re-assembling of the tree allows public visible nodes deep inside a almost completely encrypted subtree.
- It is possible to add dummy nodes to the document. The individual keys for these nodes are not included in any client-key-package. These dummy nodes allow to prevent traffic analysis attacks.
- The *big* problem of this model is performance:
  - To attach the position information to all nodes, the size of the encrypted document is increased.
  - The encryption and decryption processes are not straightforward like for subtree encryption because of the individual encryptions of the different nodes.
  - Each node needs it's own cryptographic key, so much key material is needed.

    - The flattening and re-assembling transformations cost much more processing power and memory than symmetric encryption and decryption.
- If many different recipients (with different access rights to the information) are added to the document, the performance gets better than subtree encryption. This assumes that for each recipient, the subtree encryption adds a new subtree, which blows up the document size. The larger the encrypted subtrees are and if the number of recipients grow, the system performance gets better.
- The key management allows more levels of indirection. Key management is relatively easy

Model	Subtree encryption	Server-side access control	New encryption model
Secure Complete subtrees	possible	possible	possible
Secure only Attribute values	possible (depends on proposal)	possible	possible
Leave „deep“ descendants visible	no	yes	yes
Needs trustworthy server	no	yes	no
Automation of access/encryption decisions	needs some sort of style sheets	yes	needs some sort of style sheets
Needs additional transport security	no (even complete instance can be encrypted)	yes	no
New recipients can be added without reencrypting the contents	no	only new XACL	yes

Table 2: Comparison between all models

2.3 Summary

XML as highly structured information has to be transformed by a structure-destroying process (encryption).

The presented model allows

- the flexibility of the access control models in a real encryption scenario
- traffic analysis can be prevented.

The model costs more resources:

- bandwidth to transfer,
- storage of documents,
- processing power for transformations,
- memory for complex tree operations,
- good and fast cryptographic random numbers for lots of key material.

NOTE: The current overview does not highlight the key management. I will have this till the workshop starts.

References

[Imamura] Another proposal of XML Encryption, Takeshi Imamura (Mon, Aug 14

2000)
[Simon, LaMacchia] XML Encryption strawman proposal Ed Simon and Brian LaMacchia (Wed, Aug 09 2000)
[Damiani, et.al.] Design and Implementation of an Access Control Processor for XML Documents, Ernesto Damiani, Sabrina De Capitani di Vimercati, Stefano Paraboschi and Pierangela Samarati
[Bertino, et.al.] Controlled Access and Distribution of XML Documents, E. Bertino, S. Castano, E. Ferrari, M. Mesiti
[Kudo, Hada] XML Document Security and e-Business applications, M. Kudo and S. Hada, Nov. 2000