ITEM #5: BINDING SIGNATURE TO DOCUMENT CONTENT

CASE STUDY A SUMMARY

WHAT ARE THE STEPS IN THE SIGNATURE PROCESS?

A user logs into the system by entering his/her password and answering a challenge question.   The user is then presented with the opportunity to review the document being signed and the certification statement. To sign, the user then enters his/her password again and presses the "submit" button.

WHAT CONSTITUTES THE ACTUAL SIGNATURE?

The signature is the password entered by the user.

AT WHAT POINT IN THE SIGNING/SUBMISSION PROCESS IS THE DOCUMENT ACTUALLY "LOCKED" AND WHAT IS THE LOCKING MECHANISM?

Upon receiving the submission, the system calculates a hash value for the submitted data file.

HOW ARE THE LOCKED DOCUMENT AND THE "LOCK" (E.G., THE HASH VALUE) INCORPORATED INTO THE COR?

The COR includes the submitted data file (the locked document) and the hash value (the "lock") of that file.

HOW IS THE "LOCK" PROTECTED FROM TAMPERING?

The hash value is protected from tampering in a number of ways, that include tightly controlled system access, redundant storage, and providing it back to the signer/submitter. Someone who wished to replace this hash value with a recalculated version – to hide changes in the COR – would have to defeat system access controls and access all the copies of the original, including the one in the signer's/submitter's custody.

FULL DESCRIPTION: CASE STUDY A

(Note: the description below includes relevant content extracted from an actual application.)

SYSTEM FUNCTIONS: SIGN/SUBMIT PROCESS

The signature process is a multi-step process:

1. The user must have logged onto the system by entering his account id, associated password, and correct answer to the posed challenge question. This is a pre-requisite to access any of the features of the system.
2. User is presented a screen with opportunity to review data and the appropriate certification statement.
3. User enters password and presses submit button. The ‘press’ of the button is affirmative where the button must actually be clicked or tabbed to and enter pressed to prevent accidental submissions.
4. System checks that the password matches the account and the account has the correct role to sign.
5. A confirmation number for the submission is generated.
6. An XML string of the data being submitted is generated and hashed using the SHA-256 hash algorithm. This string contains no XML header. It is only the data being submitted.
7. The full XML COR is generated by adding the XML declaration and other sections of the COR like the certification statement including signature, header and additional information sections.The additional information section contains the submittal date, confirmation number, data hash code, System version, and IP address of the user.
8. The full XML COR is hashed using the SHA-256 hash algorithm.
9. An entry is made into the submit log table with the following information:
    a. Report type
    b. Submission date and time
    c. Account number
    d. Confirmation number
    e. Program area abbreviation
    f. Data hash code (also in the COR)
    g. COR file name
    h. COR creation date
    i. Acknowledgement date
    j. System version
    k. Application server IP address
    l. Oracle server IP address
    m. COR hash code
10. The COR is stored in the copy of record table as a CLOB. ‘CLOB’ is an Oracle data type which stands for a large character field. It can hold up to 2 gigs of text.
11. The data is marked as submitted, or processed into the next system.

12. The user is sent a confirmation email with the confirmation number, data hash code, submission date and time, and directions on how to search for the COR in the System. This email is copied to up to 25 other accounts with access to the same program area ID.
13. A confirmation page is displayed with the confirmation number, data hash code, and instructions for the user to print the page.

COR Alteration Protection

The purpose of the COR signature is to provide assurances that the data was submitted through the System. COR signatures can be verified by generating the hash value of the COR and comparing it to the hash stored in the System submittal log. There are several mechanisms that allow us to detect unauthorized changes to the data. First, the person would have to have access to the database and change the data. This is extremely difficult. The person would have to have an Oracle account with the appropriate update role that was set as a default. Oracle accounts are assigned an update role if needed, but their default is the read role, not update.

There is a special procedure that must be called that temporarily allows the non-default update role to be activated. This privilege is for one Oracle call. Assuming the person found a way around this, the person would have to have knowledge of the hash algorithm used and regenerate the hash. In addition to regenerating the hash, the individual would need to modify additional tables (submittal tables which store certain data including the COR hashed code at the time of submittal) in the database to use the regenerated hash. The modification of these tables could be detected from the Oracle audit logs. DEQ enables Oracle auditing on the System database table, therefore, any time a record is modified, added, or deleted, our audit logs have record of the change. To prevent detection, the person would then have to modify the Oracle audit logs to delete the records documenting the changes. There are nightly backups of the Oracle database including the audit logs, so the changes would have to take place between the time the data was submitted and that evening when the backup was run, or the person would also have to find the backup (housed off-site and on non-changeable media/tape drives) and try to modify/re-create the backup as well.

For additional information, please see Attachment 7 which contains a description of possible scenarios for data manipulation.