

Google Summer of Code 2007 CIS

Content Integrity Service(CIS)

- Student: Jiahui Wang
- Mentor: James Rutherford
- Backup: Scott Phillips

System Design

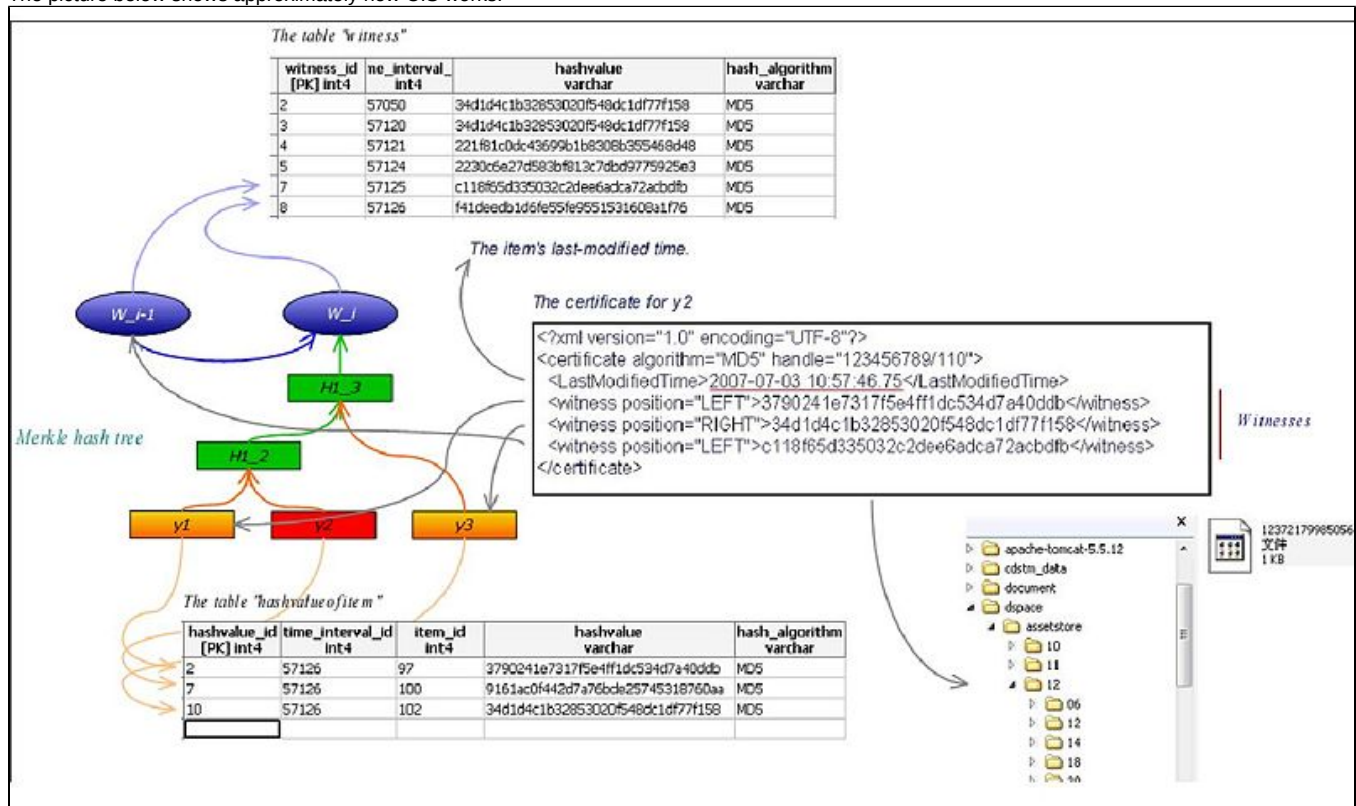
Abstraction

The goal of the service is to demonstrate that the information of the archive is authentic and has not been altered (from the point that the certificate is made). It relies on one-way hashing functions and time-stamping algorithms. It applies both transformation of the archival content and introduction of new cryptographic primitives.

Framework

The essential idea of CIS is combining hash value of document with other hash values received in the same time period to create a witness hash value. This kind of linking makes it computationally infeasible for an adversary to back-date a document, since that would entail computing hash collisions for the witness values. This technique relies only on the collision resistance properties of hash functions, and does not have any secrets or keys that need to be securely protected over extended periods of time.

The picture below shows approximately how CIS works:



As this picture illustrates, items y1, y2, and y3 were received in a time interval. The process producing y2's certificate could be described like this:

- When install an item, store its hash value in the database.
- Combine those three items' hash values and the witness hash value in the previous time interval, build a new witness for this interval and store it in the database.
- Produce the certificate for y2 and store it in the file system.

Creating and renewing certificates

Representation

A certificate includes not only the hash value of the item, but also the information of the one-way hashing algorithm and witness values. It should be like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<certificate algorithm="MD5" handle="123456789/110">
  <LastModifiedTime>2007-07-03 10:57:46.75</LastModifiedTime>
  <witness position="LEFT">3790241e7317f5e4ff1dc534d7a40ddb</witness>
  <witness position="RIGHT">34d1d4c1b32853020f548dc1df77f158</witness>
  <witness position="LEFT">f41deedb1d6fe55fe9551531608a1f76</witness>
</certificate>
```

Creating and Renewing

Create a new **bitstream** format in the table ***bitstreamformatregistry** for the certificate, with the field **internal** set to be **true**.

- Build a certificate automatically for each item when they are submitted, altered, or imported.
*The input of the renewing procedure includes the *old certificate of the item.
- The certificate should be bind to a certain hashing function, which could be appointed by the administrator.

Storage

The storage layer includes both file system and database.

Certificates

The certificate will be stored in the file system just like other bitstreams. It is treated in the same way as the bitstream **License**.

- The certificate would be given a internal-id, which is used to generate a directory for the certificate.

Database

A few of database table should be maintained, including:

- witness: table of witnesses which is published (accompanied with the corresponding time-interval).
- hash_request: hash values of the requests in the current time-interval.
 - The entries should be deleted after certificates in this time-interval have been generated.

Verification

- The service can support verification for both archival content and the certificate itself.
- Clients can download all the witness values from the service.
- The verification procedure is run in background and provides users the results.

Work in the future

Maintain ***old** certificates when generate **new** certificate.

- Make a tool generating certificates manually for items.
- Make the UI more friendly with Ajax.

See Also

- [Project abstract|<http://code.google.com/soc/dspace/appinfo.html?csaid=11628C0CA9FCEC61>]
- [A Content Integrity Service For Long-Term Digital Archives](#)
- [Hashing function APIs in JDK](#)
- [AJAX Tutorial](#)