ConCert: Content Revocation Using Certificates

Henrich C. Pöhls
University of Hamburg, Research Group Security in Distributed Systems (SVS)
poehls@informatik.uni-hamburg.de

Abstract: Content reuse on the Web is becoming even more common since the Web 2.0 "phenomenon". However, each time content is reused certain information is either completely lost (for example through excerpts) or gets harder to verify. This reduces the content's quality and thus the content's value for viewers. Among the lost information are origin, author, creation time, or if the author still endorses the content.

We propose preserving this information in a cryptographically protected content certificate (ConCert). This X.509 compatible certificate binds the content to the author's public-key. Using a hash tree we allow viewers to verify the properties, even if parts from the original content have been omitted before re-publication. The author can indicate which parts can be omitted during certificate generation in a policy. Furthermore, the author can indicate a change in his consent to the content itself or to its re-publication by revoking the content's certificate.

1 Introduction

In the "new" Web 2.0 [Rei05] the web's content is created, distributed, and displayed in different ways than before. For example content is no longer viewed or consumed directly from the author's web page or within the context that the author originally published it in. Content from different services is combined in so called "Mashups"[1]. This automated processing of content is aided by additional markup, added especially to text content, to indicate semantic meanings. Once published on the web, and thus freely available, content is subjected to further *content processing*. This processing includes, but is not limited to: harvesting, aggregation, extraction, syndication, transformation, Mashing-up, citation, and finally results in re-publication. Three parties can be distinguished: The content's author, the content processor, and the viewer who finally consumes content, processed or not. For all parties content processing introduces problems.

Two scenarios: A book price comparison service, and news articles quoting official governmental election results. A user wants further information before buying a book. Figure 1a depicts a web page with aggregated content about the book of interest. All information displayed originates from different sources, as dissected by the dotted lines. For example the book's title is provided by the site that displays the content, but all other information including each price is gathered from third parties. The user is left unaware of the true origins. Such services already exists and provide already useful services to users. Nevertheless, the user has to go to the merchant's web site manually to check the offer's validity.

[1] Mashups between 500 exisiting APIs are listed in [Pro07] in a matrix

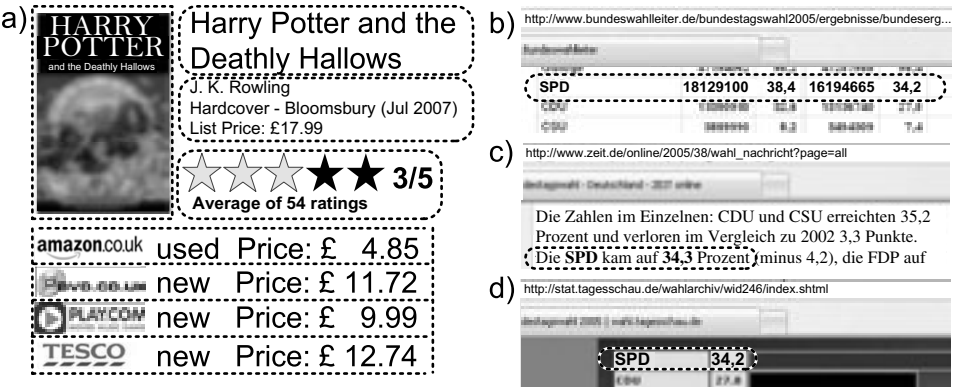


Figure 1: a) Content (ratings, prices, etc.) from different sources aggregated in new context; b) Official German election results [Bun05a]; c+d) Citation of election results in news [Die05] [ARD05]

Next time you use such a service, check how you verify the offer's validity.

In our second scenario a user wants to know the results from the German election 2005. Figure 1b shows three web pages stating them, 1b shows the official web source [Bun05a]. Imagine a user, through a search engine, has been directed to the website of a news agencies (1c or 1d). Site 1c has a hyper link pointing to the official source's domain, but not directly to 1b, the site depicted in 1d has no hyper link at all. Thus, for our user the cited results are hard to verify. To make it worse, notice the slightly higher percentage of 34,3 percent in 1c. If the user trusts the news agency to research the cited percentage correctly, it is hard to find out, that 1c correctly states only the preliminary election results [Bun05b]. Nevertheless, the values cited in 1c were correct at the time of 1c's publication.

In this paper we will further discuss the following problems of processed, re-published content: Authentication of origin, verification of compliant processing, expressing processing consent, and managing the change of consent over time. One contribution is our analysis of the problem domain (Section 3), the other contribution is our solution named ConCert (Section 3). ConCert builds on X.509 certificates and generates *author certified content* which overcomes the identified problems. Before we finally conclude (Section 7), we will briefly explain our prototype (Section 4), discuss our approach (Section 5), and contrast it with related work (Section 6).

2 Content and the Problems of processed, fragmented Content

Before explaining the problems in more detail we define what we mean by *content*. By splitting content into smaller pieces through semantic or other machine-readable annotations *micro content* is created. Micro content can be nested, included, linked to, or otherwise related to other micro content. Using the example from Figure 1a, each offer would be a micro content. Each micro content in itself is structured, as well, into *sub-properties*.

Each sub-property is addressable and identifiable. In Figure 1a, the price of a single offer would constitute such a sub-property.

Note: Content can be represented in different data formats. The problems stated, and our general approach, are not limited to a certain format, but stem from fragmentation and republication of content. So by *content* we mean any data that can be hierarchically structured into micro content and machine-readable annotated, for example XML, HTML, PDF, or layered images. But text content on the WWW is especially interesting due to the distributed nature, the often unrestricted read access, and its content re-use patterns. Browsers start to understand semantic annotations (cp. Firefox 3.0 [Faa07]) embedded in (X)HTML using Microformats [Mic06]. Another approach to the "Semantic Web" [BLHL01] builds on standards like RDFa [W3C06].

We will now look at the problems of processed and fragmented content.

Origin Authentication: Content processing modifies the content itself or at least changes the content's *context*. Context can be the original's web server, the originally surrounding content, or surrounding (X)HTML elements. The context can provide information about the origin. For example, a web server can be authenticated through means of HTTPS [DR06]. Being able to find and verify the content's author is used among other strategies to judge the credibility of information [FSD+03]. Thus, the re-publication of content under a new context reduces the ability to validate each content's original origin and reduces the content's credibility.

Compliant Processing: The processing of content is done by third-parties. For the processed content viewers have no way of automatically verifying that the processing was done in a way that preserves what the original's author wanted to be preserved. In our example case in Figure 1a different retailers might allow the extraction of prices from their website, but want their offering (so their price) bound to book's condition. We want viewers to detect policy-compliant processing. Note: This is different from detecting **non** policy-compliant processing. As we discuss in more detail in Section 5.

Consent to Processing: The original author's consent to processing and re-publication of content can not be verified. Content processing is sometimes wanted by the original authors sometimes not. If the content processing is done in consent with authors it might not cause legal problems.

For example, to be removed from Google's index and its Cache the author has to follow a certain process [Goo07]. These processes differ for different services, if they exist at all. The big publishers are looking for a solution to express their consent to content redistribution, as highlighted by the case Belgium Newspaper vs. Google [Chi06] or the Automated Content Access Protocol [ACA07]. We want viewers to differentiate author-consented processing from other forms of processing. Note: This is again different from detecting or even prohibiting **non** author-consented processing.

Especially for web content, processing (including copying) will be possible without the owner's consent. Thus, this is a weaker protection, but it does not depend on strong technical means, like digital rights management (DRM). More in our discussion on enforcement in Section 5.4.

Consent Management: The author's consent might not be infinite, processed content easily becomes out-dated. Due to the replication during processing the ability to reflect this change of consent also on all the processed content is limited. Real time fetching from the source would mitigate this, but comes at a high cost for all parties. In the scenario the citation of preliminary election results from Figure 1c the user would want to be informed that the content displayed is no longer endorsed by the source. Another valuable information for the viewer is the affirmation that the results were correctly cited from a source at the article's time of publication. The viewer is unable to gain both information automatically at the moment. Additionally to changes during lifetime, authors often have an a-priori time limit for their endorsement. For example a book store might have a special sale price for a book over the weekend only.

3 Proposed Solution: Author Certified Content

Micro content is transformed into author certified content by three steps: First the author defines the *policy* which sets out the processing rules. This policy defines a hash tree over the micro content's sub-properties, comparable to a Merkle hash tree [Mer89]. Secondly the author digitally signs a *content certificate* that binds the hash tree's root (including the micro content and the policy) to the author's public key. Finally, he publishes this content certificate alongside the content, by placing it within or around the micro content.

3.1 The Content's Policy

In the content modification policy the author defines which processing he consents to and which processing shall not be valid. Most content processing results in omission of parts of the original content, like citations or excerpts. To express the possible omissions allowed by content processors we use boolean expressions. We use `AND`, `OR`, `NOT`. Brackets `(` and `)` are used to group sub-properties. In some cases one might need the negation to describe rules for the omission of sub-properties. The sub-property identifiers contained in the boolean expression serve as variables. They have the following semantic: They are `true` if and only if a non-empty sub-property with a name equal to that of the variable is actually contained in the micro content. If the sub-property with a name equal to that of the variable can not be found within the right scope, the variable will be set to `false`.

The first step in validating author certified content is to check if the boolean expression in the policy evaluates to `true`. Only then the content has been processed in compliance to the processing rules set forth by the author. Figure 2 shows a micro content structure and a hash tree resulting from the policy `(item:fn AND price AND url) OR (item:fn AND price AND url AND condition)`. This policy allows the omission of the item's condition, but valid processing must retain the item's full name, price, and url.

As content (i.e. a web page) can contain several micro contents there can, and will be ambiguities. Ambiguities among sub-properties contained in other micro contents and the

sub-properties listed in a single policy are resolved by the content certificate's position within the hierarchical structure. Comparing it with XML signatures [ERS02] the certificate containing the signature can either be placed as an enveloping or enveloped signature. For example an enveloped content certificate is a child of the content it certifies. There can also be ambiguities within author certified content, i.e. the item and the merchant both contain a sub-property named `fn` in Figure 1. The author then needs to reference a sub-property's context by expanding it to `item:fn`. If, for any reason, there are still ambiguities, the root hash stored in the signed certificate will only match the author's original selection of sub-properties and their respective values. So if the hash tree's root is computed from different sub-property values, the validation will fail. As a result possible parsing ambiguities or author certified content placed in ambiguous context will not constitute false positive verifications.

The policy is mandatory for verifying author certified content, hence it can not be omitted. Either the policy is stored in the content certificate or it is embedded into the micro content. In both cases the policy is protected against unauthorized changes by the author's digital signature. The latter case allows compliant processing without certificate parsing, thus the content can be processed according to the policy without knowledge of how to handle certificates.

3.2 The Content's hash tree

Instead of signing the content as a whole or a hash of the whole content we first build a hash tree over the sub-properties, comparable to a Merkle hash tree [Mer89]. The policy will be used to non-ambiguously articulate the structure required to build the hash tree. As a default we use complete binary trees, like Merkle, see Figure 2b. Other trees are possible and might provide a better adoption to foreseeable content omission, but have the drawback of needing an additional representation. All the leafs of the hash tree contain hashes of all the sub-properties mentioned in the policy. A sub-property can be listed more than once inside the policy, but its value will only be hashed once. Additionally a hash of the policy itself is added to the list of hashes. The hashes are assigned to the leafs in lexicographical order from left to right. Each internal node, further up the tree, then contains a hash over the concatenation of its children, *hash(left||right)* if the node's left and right children have the values *left* and *right*. Finally the tree's root is also a hash. The root's value depends on every leaf's value.

The policy was checked to ensure author-compliant processing, as mentioned before. Now the verifier checks if the values of the remaining processed content are unchanged by reconstructing the hash tree. If values contained in sub-properties can be omitted, then a content processor that omits this data has to supply the missing information needed to re-compute the same root. The content processor adds this information in form of hash-values. Either the hash of a single omitted leaf or an inner node (if all the sub-properties attached to its children are to be omitted at once) is added. It is published with the processed author certified content as a sub-property named `authdata` identified by the node's binary number (see Figure 3). The verifier is then able to re-compute the same

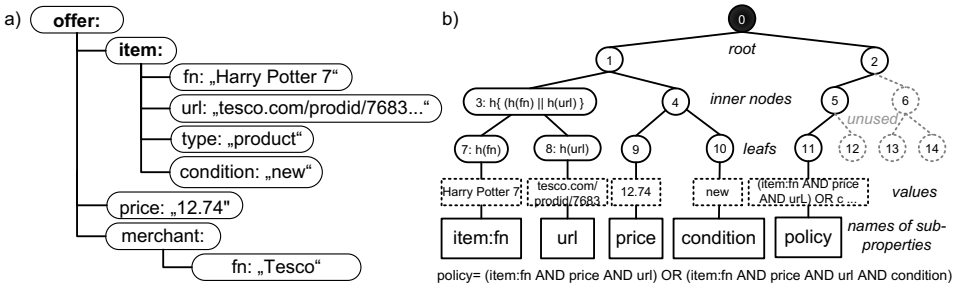


Figure 2: a) Sample micro content structure of a single offer. b) Hash tree from sample policy.

root value. Thus, by comparing the actual content's hash tree's root value with the signed value stored inside the certificate, the verifier can check the processed content's integrity.

3.3 The Content's Certificate

The hash tree's root value is bound to the author's public-key and to additional security meta data (like validity period, revocation lists, etc.) by issuing a certificate for that content. This certificate is digitally signed using the author's private-key. ConCert uses a X.509 compatible certificate [HPFS02]. To distinguish this certificate from certificates used in classical PKI we name it *content certificate*. In classical PKI a so called public-key certificate is used to certify the binding of a subject's identifier (like a real or domain name) to the subject's public key. Public-key certificates are signed by an issuer, which allows others to trust this binding.

The main difference is that our content certificate contains the content's root-hash in the subject's distinguished name (DN). The author's public-key is placed in the subject's public-key field. Thus, in contrast, we use the content certificate to bind the content's root hash to the author's public-key. We store the root hash in the subject's DN `CommonName` field (CN) and optionally use subsequent fields of the DN to hold the policy, the original URL, state and, country[2].

The content certificate is usually signed using the author's private-key. This adds the author information, like the author's name (CN), email, location into the issuer's DN of the content certificate. We allow other parties to sign the content certificate as well. Nevertheless, the author's public-key is always part of the content certificate and remains to serve as an identifier for the author. Further the field `SubjectKeyIdentifier` can help identifying the author's public-key in case of ambiguities. The rest of the certificate's fileds are used as in the classical public-key certificate.

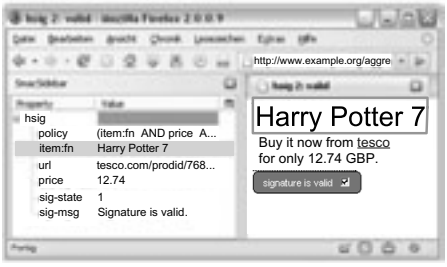
So in the end the content certificate is embedded into the micro content (see `cert` in Figure 3). In the simplest case of a self-signed content certificate, the signature is verifiable on it's own. The verifier can take the public-key needed for verifying the certificate's sig-

[2] A X.509 extension could achieve the same, but we wanted to stress the difference in the binding.

nature out of the certificate. This allows verifiers to verify that the author certified content was signed by a party knowing the private-key corresponding to the author's public-key. How verifiers establish trust in a public-key is shortly discussed in the following section 5.1, but is generally out of scope.

4 Our Prototype Implementation

We used (X)HTML content with Microformats annotation in our prototype for simplicity and to show how easily our concept can be applied to existing content. It could be implemented for example using XML as well. We implemented two Firefox browser extensions in JavaScript, one for signing already existing Microformat annotated content, the other for verification [PN07]. We also implemented a server component in PHP to allow verification, including verification of the involved certificates using Online Certificate Status Protocol (OCSP) [AMGA99] . Microformats prove useful in annotating existing content without having to break or completely redo an existing web page. We base on X.509 certificates. Note: Our use of the X.509 certificate fields is within the specification and allows us to reuse existing applications like openssl [Ope06] or Firefox. Further, it degrades gracefully if applications do not know how to interpret for example the hash in the subject's name, nor does our markup produce invalid (X)HTML or CSS. Finally, it also does not interfere with existing Microformat parsers [Kap07], as we only add new values.




```
<div id="offer1" class="offer">
    <span class="item"><h1 class="fn">Harry Potter 7</h1></span>
    Buy it now from <a class="url" href="tesco.com/prodid/768….">tesco</a>
    for only <span class="price">12.74</span> GBP.
    <div class="hsig">
      <div class="policy">(item:fn AND price AND url) OR (item:fn AN […]  </div>
      <div class="cert">  <span class="type">x509</span> <span class="value">
          -----BEGIN CERTIFICATE-----7m6NS6NCaKl42Rr24[...]</span>  </div>
      <div class="key"> <span class="type">x509</span> <span class="value">
          -----BEGIN CERTIFICATE-----3N6lftH/vok9lhqtCxQF[...]</span>  </div>
      <div class="authdata">  <span class="type">0011</span>
        <span class="value">0ea5e835c3a95c1a05d9dcd807[…]</span>  </div>
    </div>
</div>
```

CSS is used to set .policy, .cert, .key, .authdata to display:none.

Figure 3: Verification using our Firefox extension SMAC and the hsig Microformat.

5 Discussion

5.1 Authenticity of Origin, Author Tracking, Pseudonymity, and Privacy

If all the verification checks succeed, the content's origin can be linked to the process that possesses the private-key used to digitally sign the content certificate. We assume that this process can be trusted to act on behalf of an author or institution. As in other digital signature applications, signature verification builds upon trust in public-keys. Also how

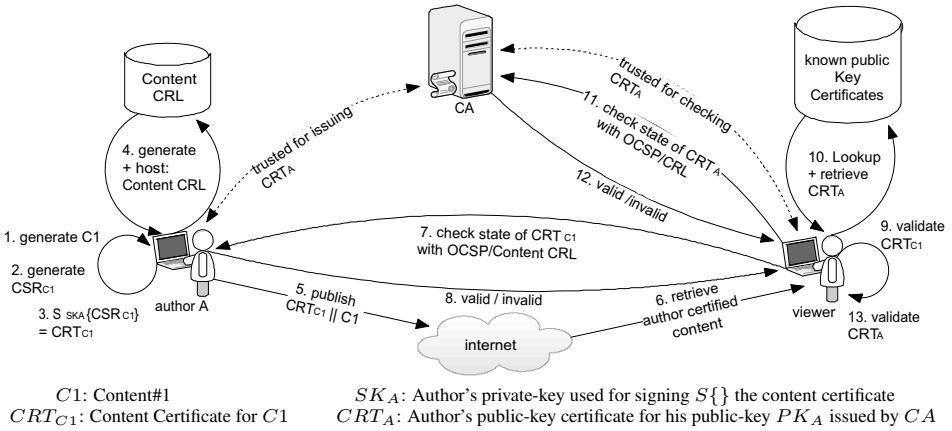


Figure 4: Self-signed content certificate and content CRL hosted by the author

viewers actually establish trust in authors is beyond the scope of this paper. Nevertheless our solution allows for several possibilities how the author's public-key could become trusted. For example an additional public-key certificate issued for the author by a trusted third-party (Fig. 4 step 11,12). Additional our scheme allows tracking all the author's contents across the boundaries of different web applications. In ConCert a public-key identifies an author, thus authors carry over reputation gained from previously published content. So viewers might have the author's public-key already on record and marked as "trusted" from previous encounters (Fig. 4 step 10).

Note: Linking authors only works if the same public-key is used. This allows authors to use different key pairs for content that they do not want to be linked with other content. They could even create a new key-pair for each content, thus using ConCert does not impose a privacy risk. Vice versa ConCert allows to claim content that was released under the "pseudonymity" of a public-key. So authors can release self-signed author certified content using a key-pair with no known binding to themselves[3]. Authorship can be claimed later by certifying content, which identifies the author, with the same key. You might even go as far as not releasing the full content, for some values you only include their hashes, allowing to re-compute the hash tree's root. To indirectly timestamp your data the hash-only content needs to be distributed in such a manner that it cannot be denied existence or removed, for example in a newsgroup [Fla07]. Then a later release of a full version can be correlated back to the author.

5.2 Processing Compliance and Integrity in the Presence of an Attacker

Processing compliance is checked by verifying that the compulsory properties set by the authors are not violated. This includes digital signature verification, but also checking

[3] Additionally no re-identifying information must be in the content certificate (i.e. CRL-URLs)

hash tree root equality, and policy compliance. The hash tree would allow an attacker to remove arbitrary (or even all) sub-property values from the content while providing the needed substitution hashes. As this is most often not desired by authors, we introduced the policy.

The author certified content's integrity is violated if any of the two checks fails. If and only if they both succeed the verifier can be sure that only sub-properties allowed to be omitted by the author were omitted, and that the remaining sub-properties' values were not modified during processing. The policy is either stored as part of the micro content or in the content certificate. Either way it is protected against unauthorized changes by a digital signature. The content's root hash is also digitally signed.

We assume that strong asymmetric cryptography and cryptographically sound hash functions are used to generate the signature. So an attacker without knowing the authors private-key can not modify or reproduce author certified content. However, there is no protection against content theft. An attacker can scrape the content, remove the authors digital signature, then do any or no change and generate author certified content bearing the attackers signature instead. However, preventing content theft is really hard once content is openly distributed over heterogeneous, decentralized networks. Nevertheless, with ConCert an attacker is not able to do non policy-compliant modifications of author certified content (bearing the author's signature) without either breaching the policy or invalidating the signature. Both are detected by the verification process.

5.3 Content Lifetime Management using existing PKI Certificate Mechanisms

Additionally to the authenticity and the integrity protection offered we propose re-using services and properties found in classical PKI for managing the binding between author and content. For this we added an X.509 certificate instead of a simple digital signature to the content. Our content certificate is compliant to the X.509 certificate structure, but holds different values and thus can be interpreted differently in the context of author certified content. We will explain now how the author can use the content certificate to indicate changes of the content during its lifetime.

In classical PKI a public-key certificate is issued and signed for a subject by a Certificate Authority (CA) (cp. PK_A in fig. 4). The certificate also contains additional information, especially a validity period and information on how to contact the issuer. If the binding between the identifier and the public-key can no longer be assured by the CA, the CA revokes (and by this invalidates) the certificate by listing it as revoked in a Certificate Revocation List (CRL) [Mer89,HPFS02]. CRLs are signed by the CA and issued in regular intervals. Verifiers could also query the revocation status of a certain certificate through OCSP. Storing public-key certificates makes offline verification possible, but for thorough verification an up-to-date status must acquired. Of course any mechanism for certificate revocation is possible.

Validity period of content certificates: For content certificates the validity period also applies, so authors can predefine when their content certificate shall expire. Again an expi-

ration of this validity period does not render the content inaccessible, it remains readable. Only the certificate is now expired. An expired content certificate is interpreted as: the author no longer vouches for the content. As an example take the preliminary election results cited in Figure 1c. Even if the content's certificate issued by the government with a validity period of two months expires, this historic author certified content is still valuable for viewers. They are still able to verify that the government once vouched for the information and that it was processed in compliance since then. Of course only as long as, through classical certificates, the government's public-key can still be trusted.

Revocation of content certificates: Our approach allows authors to change their expression of consent over time, even after content has been processed and re-published. For our implementation we reuse existing PKI tools, namely CRL [HPFS02] and OCSP. Both services allow the verifier to check if a certificate is invalid, even though its validity period has not yet expired. For both the verifier needs to be online, either at the time of verification in the case of OCSP or in certain intervals in the case of CRLs.

Authors must decide where and how to provide the revocation information before they can create author certified content as it is stored in the content certificate. The content certificate facilitates an existing X.509 extension with an URI for either the CRL distribution point or the OCSP service locator. Authors, instead of hosting their own OCSP server, might locally create their CRLs and submit them to a web service providing an OCSP for them and store the URI of this third party service. Each time the author revokes content he adds the content certificate's ID to the CRL and signs it. It is possible to provide a reason why a certificate has been revoked. Possible reasons in the X.509 standard are *superseded* or *privilege withdrawn*, which can be re-interpreted as "updated content available" or "do not re-publish".

The verifier, on the other hand, to fully verify the status of author certified content needs to contact the OCSP responder or check the latest CRL (cp. step 7 in fig. 4). Content processors and viewers alike act as verifiers, both can fully automatically check the validity of the content certificate.

This involves contacting an author defined service. Doing this every time content is viewed might raise privacy concerns ("content phones home"). In contrast to DRM systems viewers are not forced to connect in order to view the content. Applications shall leave the decision to the user, indicating that content has not yet been fully checked, but display the content as readable. The content's integrity and the compliance to processing can be verified independently.

5.4 Overhead, Gains, and Enforcement

The use of author certified content is more costly compared to normal web content. Certificate generation involves computing hash tree generation and certificate signing. Except for real-time content this only has to be done once. Our unoptimized viewing prototype took about 0.7 seconds to verify each signature found, with no impact on page rendering time. The overhead in size can be neglected with today's bandwidth.

The content certificate can be removed completely by every content processor or if still present be ignored by the viewer. We have no means to impose the enforcement at the processors nor at the viewers. Our enforcement strategy relies on the fact that all parties gain. For the viewers the benefit is obviously the automatic verification of origin, including integrity. Authors gain the ability to impose under what circumstances the verification will succeed. Not so obvious are the incentives for content processors. Two of them are: More users and a legal advantage. The latter builds upon signed license statements and the ability to distinguish third party from your own content. More users will be attracted to content processors that serve valid author certified content due to the following: Some services, especially aggregation services, depend on author content, or are author triggered [Tec07]. If content processors obey the policy, authors are much more willing to submit their work to those content processors. Additionally processors can set themselves policies on how to handle author certified content. For example a service might state in their policies that they check the content certificate's revocation status once a week and that they will not serve content with expired or revoked content certificates. Services that are not policy-compliant can be spotted by viewers as the verification will spot policy violations in the served author certified content. Policies could even be specified in machine-readable format as well (comparable to P3P [Rea01]). So all viewers regularly using the service could automatically run checks, thus processors are constantly monitored for their policy-compliance.

The restrictions known from digital rights management (DRM) systems only provide one-sided benefits [Zea07, IIM+06]. In contrast to DRM our author certified content provides added value for all parties. Achieving a win-win situation for all parties we envision an enforcement by the "power of crowds", as more and more users will prefer being presented processed content of which the credibility can still be judged.

6 Related Work and existing Implementations

Content revocation is often mentioned in the context of DRM systems (e.g. AACS [IIM+06]). But our form of content revocation does not limit or restrict the access to the content itself. Rather we revoke additional relationship guarantees, in our case the relationship between the author and the content is revoked. Secondly AACS does not use X.509 compliant mechanisms. The concept of hash trees is also already used in content distribution (e.g. P2P systems). But in P2P systems the hash tree's root is not cryptographically protected. Several web services strive to solve the problem, closes to our approach is a service called FindMeOn [Van06]. Finally an existing mechanism to verify the original server's authenticity is through Transport Layer Security (TLS) over HTTP, known as HTTPS. TLS's goal is to secure connections, thus the authenticity is not preserved after the connection has ended. We thoroughly analysed more existing web services finding that none of them has all the desired properties. The full details are beyond the scope of this paper.

The idea of signing web content is not new as a mailing list discussion [RM02] or work done in the area of caching signed web content [CW02] from 2002 shows. However, recently, based on our technical report [PÖ7], Quasthoff et al. [QSM07], also argued

that HTTPS can not solve the problem. They embedded standard XML signatures into XHTML. While this allows origin authentication it neither allows the content's binding to the author to have a validity period or to be revoked, nor it tackles the problem that web content is often only partially redistributed.

Work done by Devanbu et al. [DGK+01] allows to verify the answer to a selection query over a XML document. They also use a Merkle hash tree to allow omission of parts of the XML document not asked-for in the query. Clients obtain the hash tree root from the authors in a secure way and verify if the returned parts of the document are complete and correct. Bertino et al. [EBE+04] in 2004 and more recently in 2005 Carminati et al. [CFB05] show how XML data can be protected without trustworthy publishers. Their approach additionally employs confidentiality protection through encryption. Our approach enhances these two with an author-defined policy to restrict what content can be omitted rather than allowing any queries to be valid. Our focus is on Internet content created for publication, as such confidentiality plays a minor role.

ConCert's main difference from the existing approaches, is the ability to revoke the authorship attribution. Still we use as much existing mechanisms to ease adoption as much as possible. To the best of our knowledge no one has yet proposed the use of certificates and other PKI mechanism to add attributes to web content and to allow revocation after it has been processed and re-published by third-parties.

7 Conclusion

Our proposed *author certified content* allows viewers to verify the content's origin through the author's signature. Using hash trees we further allow the verification to succeed even after certain parts are removed (i.e. during citation) as long as the correct substitution information is added to recreate the hash tree. What processing can be applied to the content is securely stored in a policy. Lending the mechanisms of certificates and certificate revocation (especially CRL and OCSP) from PKI allows us to indicate the premature end of the bond between the content and its author. The result of this revocation remains visible for viewers even after the content got spread, as long as only valid content processing was applied. The content's accessibility is not affected.

Even though the technologies we used are well understood and already embedded in today's browsers and servers, the origin verification and the revocation was not possible before. With our prototype implementation we demonstrate that the concept is suitable for Web content and it is ready for today's (X)HTML based Web. Nevertheless, our general concept is applicable to all forms of content that can store cryptographically secured meta-data.

New services can be build upon author certified content and already existing services can inherit the new possibilities. As author certified content contains no service specific identifiers and builds on open processes it is applicable across the boundaries of the Web. For example, using an existing Mashup, viewers can verify each author certified content, without a change of the Mashup.

References

[ACA07] ACAP. ACAP website. www.the-acap.org, Oct. 2007.

[AMGA99] R. Ankney, A. Malpani, S. Galperin, and C. Adams. RFC 2560 - X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP. www.ietf.org/rfc/rfc2560.txt, June 1999.

[ARD05] ARD. Bundestagswahl 2005. stat.tagesschau.de/wahlarchiv/wid246/index.shtml, Oct. 2005.

[BLHL01] T. Berners-Lee, J. Handler, and O. Lassila. The semantic web. *Scientific American*, May 2001.

[Bun05a] Der Bundeswahlleiter. Endgültiges Ergebnis der Bundestagswahl 2005. http://www.bundeswahlleiter.de/bundestagswahl2005/ergebnisse/bundesergebnisse/b_tabelle_99.html, Oct. 2005.

[Bun05b] Der Bundeswahlleiter. Vorläufiges amtliches Ergebnis der Bundestagswahl 2005. http://www.bundeswahlleiter.de/bundestagswahl2005/presse/pd360211.htmll, Sept. 2005.

[CFB05] B. Carminati, E. Ferrari, and E. Bertino. Securing XML data in third-party distribution systems. In *Proceedings of 14th ACM CIKM*, pages 99–106, 2005.

[Chi06] Chilling Effects Clearinghouse. English-language translation of Belgian Court ruling. www.chillingeffects.org/international/notice.cgi?NoticeID=5133, Sept. 2006.

[CW02] Chi-Hung Chi and Yin Wu. An XML-Based Data Integrity Service Model for Web Intermediaries. In *International Workshop on Web Content Caching and Distribution (WCW)*, 2002.

[DGK+01] P. Devanbu, M. Gertz, A. Kwong, C. Martel, G. Nuckolls, and S. Stubblebine. Flexible authentication of XML documents. In *8th ACM Conf. on Computer and Comm. Security*, 2001.

[Die05] Die Zeit. Farbenspiele in Berlin. images.zeit.de/text/online/2005/38/wahl_nachricht, Oct. 2005.

[DR06] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.1. RFC 4346 (Proposed Standard), April 2006. Updated by RFC 4366.

[EBE+04] E.Bertino, B.Carminati, E.Ferrari, B.Thuraisingham, and A.Gupta. Selective and Authentic Third-Party Distribution of XML Documents. *IEEE TKDE*, 16:1263–1278, 2004.

[ERS02] Eastlake, Reagle, and Solo. XML-Signature Syntax and Processing. W3C Recommendation. www.w3.org/TR/xmldsig-core/, Feb. 2002.

[Faa07] Alex Faaborg. The User Interface of Microformat Detection. blog.mozilla.com/faaborg/2007/02/04/, February 2007.

[Fla07] Halvar Flake. Dailydave: Some Sums. seclists.org/dailydave/2007/q1/0136.html, Feb 2007.

[FSD+03] B. J. Fogg, Cathy Soohoo, David R. Danielson, Leslie Marable, Julianne Stanford, and Ellen R. Tauber. How do users evaluate the credibility of Web sites?: a study with over 2,500 participants. In *DUX '03: Proceedings of the 2003 conference on Designing for user experiences*, pages 1–15, New York, NY, USA, 2003. ACM Press.

[Goo07] Google. Prevent or remove cached pages. `www.google.com/support/webmasters/bin/answer.py?answer=35306`, Apr 2007.

[HPFS02] R. Housley, W. Polk, W. Ford, and D. Solo. RFC 3280 - Internet X.509 PKI Certificate and Certificate Revocation List (CRL) Profile, Apr. 2002.

[IIM+06] Intel, IBM, Matsushita, Microsoft, Sony, Toshiba, Walt Disney, and Warner Bros. Advanced Access Content System (AACS) Pre-recorded Video Book Rev.0.91. `www.aacsla.com/specifications/specs091/AACS_Spec_Prerecorded_0.91.pdf`, Feb. 2006.

[Kap07] Mike Kaply. Operator. `www.kaply.com/weblog/operator/`, Oct. 2007.

[Mer89] R.C Merkle. A Certified Digital Signature. In *Advances in Cryptology*, 1989.

[Mic06] Microformats. website. `www.microformats.org`, Aug. 2006.

[Ope06] OpenSSL Project. *OpenSSL*. OpenSSL, Dec 2006.

[PÖ7] Henrich C. Pöhls. Authenticity and Revocation of Web Content using Signed Microformats and PKI. Technical Report B-276-07, University of Hamburg, Department of Informatics, Hamburg, Germany, February 2007.

[PN07] Henrich C. Pöhls and Henrik Niklaus. SMAC. `http://www.informatik.uni-hamburg.de/SVS/personnel/henrich/hsig.php`, Oct. 2007.

[Pro07] ProgrammableWeb.com. Mashup Matrix. `programmableweb.com/matrix`, Aug 2007.

[QSM07] M. Quasthoff, H. Sack, and Ch. Meinel. Why HTTPS is Not Enough – A Signature-Based Architecture for Trusted Content on the Social Web. In *IEEE / WIC /ACM Int. Conf. on Web Intelligence*, 2007.

[Rea01] Joseph Reagle. P3P Assurance Signature Profile. `www.w3.org/TR/xmldsig-p3p-profile`, Feb. 2001.

[Rei05] T. O' Reilly. What Is Web 2.0. `www.oreillynet.com/lpt/a/6228`, Sept. 2005.

[RM02] Doug Ransom and Peter V. Mikhalenko. Discussion on "XHTML adoption curve". `lists.xml.org/archives/xml-dev/200204/msg00559.html`, Apr. 2002.

[Tec07] Technorati. Pingerati. `pingerati.net`, Feb 2007.

[Van06] Jonathan Vanasco. FindMeOn. `findmeon.com`, Sep. 2006.

[W3C06] W3C. RDFa Primer. `www.w3.org/TR/xhtml-rdfa-primer`, May 2006.

[Zea07] State Services Commission New Zealand. New Zealand E-government Programme Glossary. `www.e.govt.nz/policy/tc-and-drm/principles-policies-06/chapter7.html`, Apr 2007.