Andreas Heuer, Frank Losemann, Christoph Meinel:
Logging and Signing Document Transfers on the WWW - A Trusted Third Party Gateway
in Proceedings of the First International Conference on Web Information Systems Engineering (WISE 2000), vol. 1, IEEE Press, Hong Kong, China, pp. 146 - 152, 6, 2000. ISBN: 0-7695-0577-5.

Logging and Signing Document-Transfers on the WWW- A Trusted Third Party Gateway

A.L. Heuer, F. Losemann, C. Meinel
{Heuer,Losemann,Meinel}@ti.fhg.de
Institute for Telematics
Bahnhofsstr. 30-32, 54292 Trier, Germany

Abstract

In this work we discuss a service that aims to make quoting of online documents, „web contents“ easy and provable. For that reason we report the conception of a gateway that works as a Trusted Third Party (TTP) service which is based on a public key infrastructure (PKI). The developed service consists of the signing of any data-transmission that was done via the TTP-gateway. After the data-transfer a set of data can be requested from the used gateway that is signed with the TTP-gateways private key. This signed set of data contains for each request that was processed by the gateway at least three components. Those are the request from the client, the reply from the server and finally the signature of the (TTP-)server. Storing this signed data the recipient at the client side can provide it to other parties suitable for a latter verification of the data-transfer. The TTP-server generates automatically verifiable statements of the kind „this request resulted in that response“. Now anyone that trusts the chosen TTP-gateways statements will be able to verify the data-transfer by the use of the trusted third parties certified public key. Furthermore this paper describes a prototype implementation of such a service using HTTP. Finally a possible employment of the TTP-gateway is discussed.

Introduction

This paper focuses on a very common problem on the World Wide Web(WWW). With the WWW being a highly dynamic medium it is quite obvious that with time going by most resources are changed or finally removed at all. If a document is removed or renamed this often results in a broken link and hinders any further access to the document. If there is no archive provided the document is lost for any further use. Eventually more frustration arises from a changed document. While a substantial change probably will be recognized by anybody that referenced that document minor changes in details are difficult to be recognized without the help of difference engines [1, 2, 3]. Since a removed or modified document is worthless as a reference some solution to the broken link problem is necessary. Although there are several efforts [4, 5, 6, 7] and therefore several mechanism, e.g. removal of the link from all referencing document, are employed to solve this, they still have not been stablished. For example the reading of most online References of the former WWW conferences is impossible since a huge number of links already expired.

If it is not possible to solve the problem of broken links in general it might be helpful to maintain a local copy of a online reference. This can be a personal solution, but is not practicable in common, since the local document lacks any authenticity. Independent of the type of connection, simple HTTP[8] or a more secure derivative, e.g. HTTPS [9] or S HTTP [10], the client has no means to prove later, that it indeed received that certain document from the server and that it has not been modified in any way. So the special problem is how to get an authenticated document as a substitute for a online reference, if the owner of the reference, e.g. the author or the web-server itself, does not provide such a service. In common the problem with the actual client server architecture of the internet is, that data-transfers itself are not commonly verifiable afterwards.

The approach discussed in this paper focuses on this fact. We suggest a new type of service that can be used by anybody without modification of neither the server nor the client. This service provides for any client the possibility to get some trustable proof for the data transfer it handled. While the concept explained in this paper might be applicable for most data transfer protocols that support a kind of gateway, it is prototypal developed for the use of HTTP.

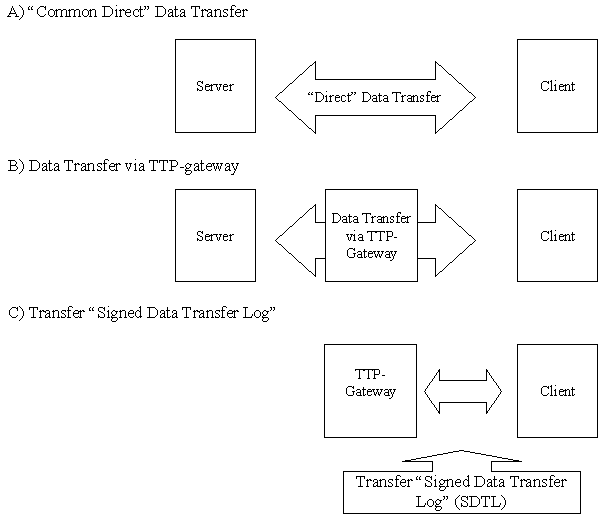
TTP-Gateway Concept

In this section we describe the concept we developed to offer a provable data transfer. It is based directly on the existence of a trusted third party (TTP) that signs any data transfer it handles. Dependent on the field of employment of this concept different requirement for the TTP will arise. Given the problem of online references, e.g. a digital library, could be accepted as a TTP. As well external proxies of an intranet could offer a service for use within the intranet. The TTP we focus on has to care for two services. On the one hand there is the functionality of a gateway. The TTP has to care for handling of data transfers between any client and server.

On the other hand a service must be provided that protects the TTP output from tampering preserving authenticity. Furthermore a mechanism for later data validation is needed. This could be achieved by signing [11] the results of the TTP-gateway including a statement, that describes the service provided, i.e. how the statement granted by the TTP-gateway is to be interpreted (policy). To provide this TTP-gateway service to the whole internet community the TTP-gateway should be integrated in a commonly accepted Public Key Infrastructure (PKI), e.g. the PKIX Internet X.509 Public Key Infrastructure as described in PKIX Charter [12]. Making use of this infrastructure its means for data validation [13] as well as certificate formats [14] are determined. The integration of the TTP-gateway into this infrastructure can be handled as usually by any commonly accepted internet certifying authority (CA).

While the client usually connects directly to a server (Picture 1A), the concept of the TTP-gateway requires that the client does not connect itself directly to the server but transfers its data via the gateway (Picture 1B). Since the complete communication between the client and the server is handled by the TTP-gateway, it is able to log the whole session. Of course one would only allow a trusted party this freedom. In this case one has to trust, that the data will not be available to anybody else, until one decides to provide it oneself.

But trust is required in an other, broader sense as well. After the data transfers via the TTP-gateway between the servers and the client are finished, the client can request from the TTP-gateway a log of the data transfer (Picture 1C). In order to allow a latter validation by interested parties, the TTP-gateway has to sign the data transfer it handled with its private key. So the Signed Data Transfer Log (SDTL), which the client can receive for either single requests or any set of requests, enables the user to prove to anybody that trusts in the party that runs the gateway, what data he/she did receive on a certain requests. Therefore the SDTL has to contain several components, at least three. The first component is the request the client did send to the server. With this component it is possible to validate which request did result in the received document in focus. In the case of HTTP the request headers are essential, since different browsers or language preferences may result in different replies. At second the SDTL has to contain the server reply to the request. In the case of HTTP this reply splits into two parts, the response header and the reply data itself. Having HTTP in mind, the response header contains information about the reply data, e.g. the content-length. Now anybody that has access to the SDTL can read the request, the reply and therefore the document that was transferred.



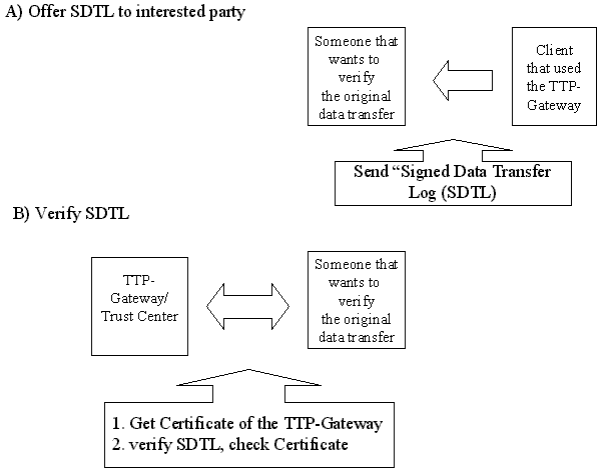
Picture 1 In order to get a provable data transfer the client has to switch from a direct connection to the server (A) to a connection that is handled by the TTP-gateway (B). In a last step the client requests from the TTP-gateway the Signed Data Transfer Log (C), a file that contains a signed prove for the data transfer handled by the gateway.

Finally the third component contains the signature of the client-request-component and the reply component sent by the server. In order to make the mechanism work quite stand alone the signature-component may also contain the certificate of the TTP that did the signing. At last a HTML-file containing instructions how to interpret and verify the given signatures for the request/response can be approved.

With a SDTL created in the described format the client is now able to prove to anybody that trusts in the party running the TTP-gateway, that indeed a request, as it is contained in the SDTL did result in the response, that is included in the SDTL as well. The document validation process requires at least three steps. In the first step the one who wants to check the SDTL has to get this file (Picture 2A). The SDTL can be provided either by the original client itself or any other party.

In the second step, the validating party has to get and validate the certificate of the TTP-Gateway. Several solutions for the certificate transfer are possible, either a secure connection (e.g. HTTPS) can be used to request

the certificate from the TTP directly, or a Data Validation Server (DVCS), see [13] can be involved. With the knowledge, that the certificate is indeed assigned to the TTP running the TTP-gateway the content of the SDTL can be validated (Picture 2B). Of course it is also possible to implement a public service the handles the complete validation of a SDTL, including validation of the certificate and the SDTL's contents.



Picture 2 When the validating party received the SDTL (A), the Validation of the SDTL includes at least two steps(B). The certificate of the TTP has to be validated as well as the content of the SDTL.

Given the special case that someone wants to use an online reference (available via HTTP) in his/her paper that is to be published online, it is now possible to provide along with the hyperlink to that document a SDTL of a former retrieval of the document by the author. The SDTL can be handled as any image data that is linked in the paper. When that paper is published the SDTL will still be available along with the paper, when the original online document was already removed or modified in between. So this enables any user that trusts in the TTP to validate independently the content of the reference. Trusting in the SDTL of the online reference, there is an added value given by the SDTL even if the online reference itself is still available. This value arises from the ability to find out, if the document was modified since the reference was made.

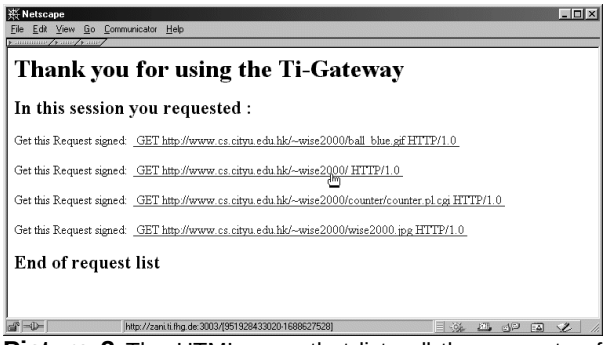
HTTP-Prototype Implementation

In this section a prototype implementation of the TTP-gateway in JAVA for the use with the commonly used HTTP is described. Although this programming language has disadvantages concerning the performance, we chose it for the prototype, since the given security tools [15] and the JAVA security API [16] allows a quite easy handling of security related problems. Furthermore we decided to use the signed jar-files [17,18] as SDTLs. This decision was based on the following reasons: At first jar files are prepared to be signed, since they are used to transport trusted mobile executables from servers to clients. The second reason is the availability of an compression algorithm, since jar files are based on the zip file format [19]. At third there exists with the jarsigner [15] at least one commonly available tool for an immediate verification of the content and signature of a jar file. Although the handling of this command-line tool is quite poor, the use in a prototype is appropriate.

The prototype implementation consists of a multi-threaded server program that listens on a given port for requests. Clients are distinguished by their ip-addresses in order to get a kind of session information that lasts longer than stateless HTTP request. Problems with data-privacy arise, if several users share the same ip-address. Therefore the prototype can not be used for security relevant browsing sessions.

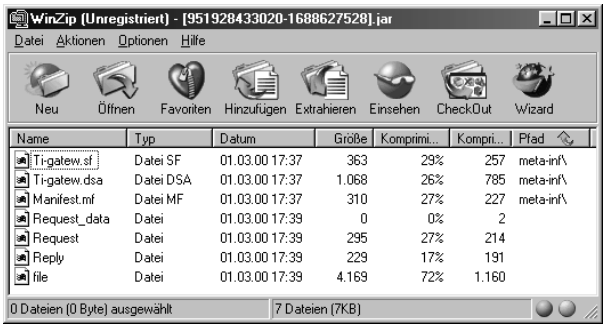
For each client a kind of cache is build in the gateway to store the data of the recent (a time period can be defined) browsing session. This stored data includes the request header, the request-data (POST-method), the reply header and finally the reply data, respectively the file received by the client.

If the user finished his/her browsing session via the gateway, he/she can connect via HTTP directly to the TTP-gateway. Based on the ip-address of the client all requests of the recent browsing session are now listed on a HTML-page (Picture 3). This allows the user to decide, which requests he/she needs to be signed. As a reply to the user request for a SDTL he/she will receive the SDTL in form of a signed jar file. Although its name is a little bit cryptic at the moment (unique id) this single, compressed file contains anything required for validation of the complete data transfer of the concerning HTTP-request.



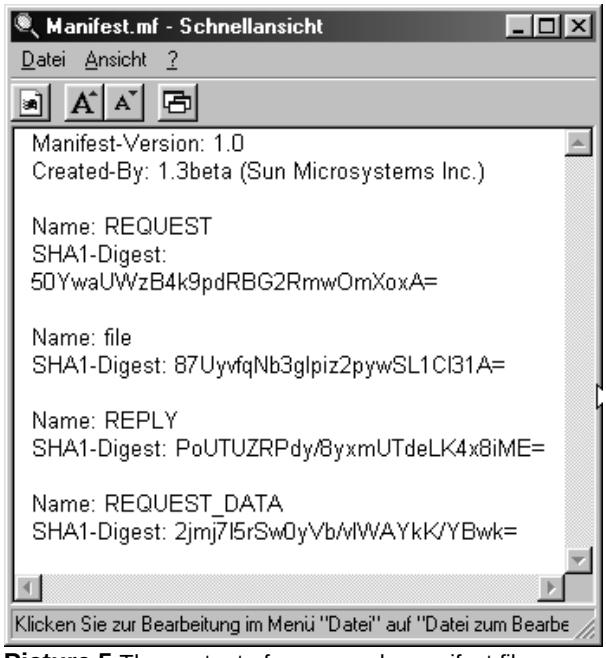
Picture 3 The HTML-page that lists all the requests of the last "session". Each request is handled separately in order to provide high flexibility to the user. A mechanism that combines all request (e.g. HTML-page, with the bound images inside of that page) of a certain document. By click on a link, the SDTL will be created and sent to the user.

In detail the signed jar file consists, in the case of a single HTTP-transfer signed, of seven files (Picture 4).



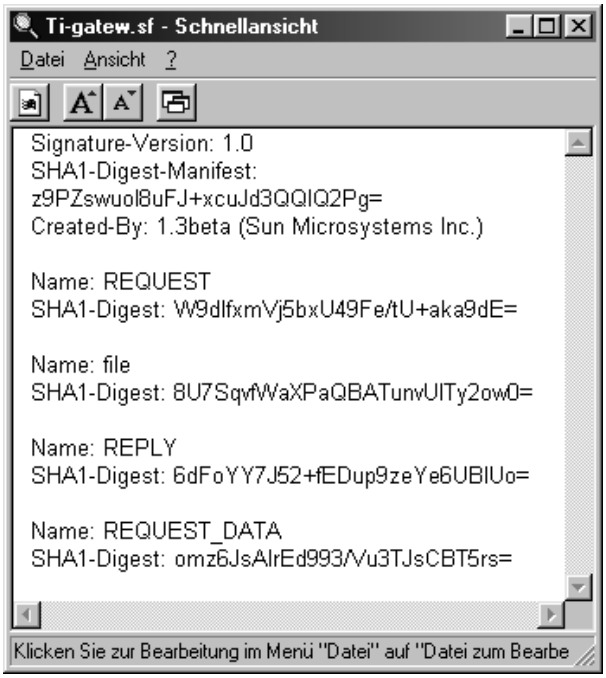
Picture 4 Since the jar-file is based on zip it can be opened with programs like winzip[*20*] in order to view the content. As one can see, there are seven files included. Those are the Request (header), the Request_data (empty), the Reply (header) and the file, that is the document, one is interested in. Furthermore there is the Manifest of the jar, the signature file (in this case labeled Ti-gatew.sf, since this was the selected alias of the certificate owner) and finally the signature block (Ti-gatew.dsa)

Of those files four "contain" the data transfer. Three files, stored in a subdirectory (meta-inf), are required in a signed jar file[21]. The first of those later files is the manifest file.



Picture 5 The content of an example manifest file.

It contains for each of the four member files a message digest computed e.g. with the MD5 [22]or SHA[23] algorithm. The second of those files is the signature file (Picture 6).



Picture 6 The content of an example signature file.

It contains digest entries for the archive's files similar to the digest-value entries in the manifest. While the digest values in the manifest are computed from the files themselves, the digest values in the signature file are computed from the corresponding entries in the manifest. Since the signature file is compatible with version 1.2 of the Java-platform it also contains a digest value for the entire manifest. Finally the third of those files is the signature block, a not human-readable file in the PKCS#7 format [24]. It contains two elements that are essential for the verification, the digital signature for the jar file and the signers certificate. The digital signature is generated with the trusted third parties private key [conforming to PKCS#7]. The certificate contains the public key of the trusted third party. Therefore one needs only to provide that signed jar-file to enable others to validate the data transfer.

Two different modes for the request for signatures are available right now. In the single request mode each request of the browsing session can be requested in a separate file. This allows the user a selection of the required data. The multi request mode works slightly different. It offers the user a single signed jar file containing all requests with their related data. This mode is very effective, if there was a document browsed via the gateway that contains many other important sub-components e.g. images. In that case the jar file would contain additionally to the document itself the related components, too. Of course this mode is also very useful when browsing documents split into several parts.

But also all components of the document are afterwards available in the SDTL, the links in the document appear broken, since no modification of the content of the SDTL was done. Probably a kind of SDTL-Viewer can be built that can handle such link transformations.

Given the scenario a user reading a document that contains a online reference as well as a link to a corresponding signed jar-file. Then there are two potentialities. Either the online reference is still available or it cannot be loaded anymore. In both cases it would be a good idea of the user to load the signed jar-file anyway. In the first case the user can compare the actual document located at the online-reference with the document that the author had in mind when using that reference. In the other case the user is able to read the document regardless to the invalid original online reference.

If the user did load the signed jar-file there are several steps to perform in order to assure the content. The first step is the verification of the signed jar-file. The verification of the signed jar file is simple, given the fact that the jarsigner -tool, distributed with the common JDK [25], is commonly available. Running the program with the -verify -certs options, will start the verification and also display the certificate of the signer (the trusted third party).

```
jarsigner -verify -certs -verbose [9519284330 20-1688627528].jar

        310 Fri Nov 30 17:37:08 GMT+01:00 1979 META-INF/MANIFEST.MF
        363 Fri Nov 30 17:37:44 GMT+01:00 1979 META-INF/TI-GATEW.SF
       1068 Fri Nov 30 17:37:44 GMT+01:00 1979 META-INF/TI-GATEW.DSA
sm      295 Fri Nov 30 17:37:06 GMT+01:00 1979 REQUEST

      X.509, CN=Ti-Gateway, OU=Security, O=Institute for Telematics, L=Trier, ST=RP, C=de

sm     4169 Fri Nov 30 17:37:06 GMT+01:00 1979 file

      X.509, CN=Ti-Gateway, OU=Security, O=Institute for Telematics, L=Trier, ST=RP, C=de

 m        0 Fri Nov 30 17:37:06 GMT+01:00 1979 REQUEST_DATA
sm      229 Fri Nov 30 17:37:06 GMT+01:00 1979 REPLY

      X.509, CN=Ti-Gateway, OU=Security, O=Institute for Telematics, L=Trier, ST=RP, C=de


  s = signature was verified
  m = entry is listed in manifest
  k = at least one certificate was found in keystore
  i = at least one certificate was found in identity scope

jar verified.
```

Picture 7 A screenshot after the execution of the jarsigner with the -verify option for the example jar.

Supposed the verification was successful in the next step the user should validate the certificate provided inside the jar-file. Validated that the jar-file was indeed signed by the TTP-gateway the user now can concentrate on the content of that file. The user can extract the request information as well as the document in focus. Since the request header as well as the reply header are human readable the verifier can reproduce the communication that brought up the document that was in focus of the online reference.

Attacks against the gateway

Of course question of security and possible attacks are very relevant, when there is a PKI involved. But obviously this field of interest must be discussed in more detail than it is possible in this paper. Therefore we give here only a brief statement about the attacks that can be expected concerning the described concept. Following the design of the TTP-gateway it is obvious that somebody using the gateway should receive the same data from the web as somebody not using the gateway. So if there happens any attack of the kind man in the middle, it would have happened without the gateway anyway. For a man in the middle attack there are two working points. The first is located between gateway and client. This one can be neglected since any modification of the signed jar file will result in a failure of the verification. The second point of attack is located between the gateway and the web-server serving the requested document. This attack can only be hindered if a secure connection between that server and the gateway is used. Since the overhead given with the handling of a secure connection is not inevitable, the implementation of such a feature depends on the security requirements of the users. Anyway the number of web-sites that can be accessed with a secure connection is remarkably small against the number of servers using unsecured connections.

Other attacks against the TTP-gateway will aim for the private key used to sign the data-transfer. Attacks of that kind can happen to everyone employing public key infrastructures. Therefore common security concepts have to be implemented to secure the gateway.

Possible Employment of the TTP-gateway

One can imagine several possible fields of employment for the Signing Transfer Service (STS) provided by the TTP-gateway. In particular there are two kinds of service providers. The first is the conventional service of a gateway as described in this paper. But there is a second one, that is a special constellation. This second kind is a web-server that itself provides the signing functionality for each request handled at a certain port. While in the first case the STS is independent of any special web-space, in the second case the web-site and the STS are more or less integrated.

The STS as an independent solution can be provided by any trusted party. Employment candidates are in this case for example search engines, web-portals, or institutional sites as universities. They would be trusted by a wide variety of parties and therefore be prepared for a widespread use.

Institutions as well as companies can provide the STS as an integrated service in order to give more reliability to their customers or employees. This is especially

interesting since the combination of the web-server and the STS avoids the man in the middle attack. Therefore the mentioned parties can provide for any document delivered an additional SDTL. E.g. a internet shop could offer for each handled purchase a signed proof in form of a SDTL to the customer.

While such purchase proofs are to be for private use only they will not interfere with any copy right. This is different, if one publishes along with an online document the referenced documents as SDTLs. In this case the copy right of the original authors has to be taken into consideration. So either an agreement of those authors is available or the SDTL can only be stored for personal use, e.g. to prove the correctness of a reference.

Finally again independent of this difficulty the STS might be very interesting for some companies, as they might employ it in order to offer their employees a reliable framework for internal use inside the intranet.

Outlook

With the prototype of the TTP-gateway an example implementation of the STS concept is for testing available. Using jar-files as containers this simple approach guaranties easy implementation and broad availability of the validation tools. While the prototype is for testing purposes a good solution, there are several drawbacks on the client side. The tools that allow the validation of the SDTL are command-line based and the complete validation process, including the import of the TTP's certificate, is quite circumstantial. Efforts will be undertaken to develop a client program with an adequate Graphical User Interface that encapsulates the complete validation process. Required features are easy import of certificates, simple verification of SDTL with a graphical user-interface and probably a kind of SDTL management. Of course it would be easier for the users if the validation process is located on a server. A simple upload via a secure connection could transfer the file in question to the validating server. This method would require additional trust in the validation server.

Independent of the usability of the prototype there are several other features to be implemented in future developments. The feature needed probably at most is the provision of a secure connection from the TTP-gateway to the server. This would decrease the probability of the man in the middle attack described above. If confidentiality or anonymity are desired for SDTLs, a secure connection from the gateway to the client could be used too. Such a secure connection would furthermore provide something like a session that could be tracked inside the gateway independent of the ip-address as it happens at the actual prototype.

Other features possible, are a time-stamping-service and probably a persistence service. While the time-stamping-service certifies the time for each SDTL and therefore allows the classification into a historical context, a persistence service would care for the long-term availability of the SDTLs. Those additional features could be provided in connection with external services, according to different quality of service levels, up to the user's choice.

Related Work

In the context of signed documents there are two initiatives to mention. On the one hand there is xml-sig [26]. The specification of XML-Signature provides a mechanism for applying digital signatures to XML documents and other Internet resources and encoding those signatures as XML. Its structure allows for both embedded and detached signatures. On the other hand the Digital Signed Label Architecture [27] is to mention. The SDTLs described in this paper are similar to XML-Signatures or Digital Signed Label (DSL) in the sense, that they are assembled to make a signed assertion about an information. While the concept of the TTP-gateway is independent of the format used for the SDTL, it may be interesting to discuss the use of XML-Signature or DSLs as format for the SDTL.

Another program providing services on a similar field is the SSL Proxy [28] that can be used to transfer data with an secure connection even if, for example the original web-server, does not support a encrypted protocol (HTTPS). Indeed this program would be a great complement for the TTP-Gateway, since it provides secure connections between clients and servers. Of course we are aware, there are other services available on the WWW that offer similar features.

E.g. there are certified delivery services such as AuthentiX (www.true-email.co.il) that offer proof of what was delivered (downloaded) via Web-based email. Although they provide a similar functionality there is a big difference to the STS, since the STS does not require any modification of an existing service. It can just simply be plugged into the communication between client and web-server. Finally the combination of several independent services in the appropriate, complicated way would obviously achieve results providing the same statements. The most important of those services to mention are time stamping services[29] and electronic notaries[31]. Time stamping services bind a statement to a given point in time. Electronic notary services focus on the authenticity and persistence of certified statements themselves. Our service links an URI or, more precise a protocol request, to a protocol response at a given time. The time included in the TTP-gateways statement could of course be additionally improved by obtaining and adding a time stamp from a dedicated service to the response. The requirement for including all protocol request headers in the SDTL arises e.g. from the existence of web-sites that adapt to different language preferences or browser identification strings sent by browsers. In the case of HTTP a web-server

implementing content negotiation [31] this could result in different documents sent as response to a request of the same URL.

Conclusion

With the concept of the TTP-gateway a service is developed that provides a mechanism for the signing of data-transfers. Since any data-transfer can be signed by simply directing it through any TTP-gateway the use is multifarious. After choosing a convenient provider of such a service anybody can use the mechanism to get a verifiable documentation about certain data-transfers. This rises the reliability of the resources of the world wide web a lot. With the simple prototype implementation of the service for HTTP the feasibility of the concept is proven. Using the signed jar files to be verified by the commonly available jar-signer tool increases the practicability of the technique. Since the service is based on a public key infrastructure it depends strongly on the availability of trusted parties that will provide this kind of service to their customers. We think, with our service we offer a solution to some of the problems in focus of the article "Intellectual Preservation: Electronic Preservation of the Third Kind" [32].

References

[1] Fred Douglis, Thomas Ball and Yih-Farn Chen ,WebGUIDE: Querying and Navigating Changes in Web Repositories, Fifth International World Wide Web Conference, May 6-10, 1996, Paris, France, http://www5conf.inria.fr/fich_html/papers/P38/Overview.html

[2] Thomas Ball and Fred Douglis. An internet difference engine and its applications. In Proceedings of 1996 COMPCON, February 1996, pp. 71-76.

[3]Fred Douglis and Thomas Ball . Tracking and viewing changes on the web. In Proceedings of 1996 USENIX Technical Conference, January 1996

[4] The PURL Homepage. URL: http://purl.oclc.org/

[5] David Ingham, Steve Caughey, Mark Little, Fixing the "Broken-Link" Problem: The W3Objects Approach, Fifth International World Wide Web Conference(May 6-10, 1996, Paris, France), http://www5conf.inria.fr/fich_html/papers/P32/Overview.html

[6] A. Aimar et al., "WebLinker, A Tool for Managing WWW cross-references," Computer Networks and ISDN Systems, Vol. 28 No. 1&2; Selected Papers from the Second World Wide Web Conference, December 1995

[7] F. Kappe, K. Andrews, and H. Maurer, "The Hyper-G Network Information System," J.UCS Vol. 1, No. 4 (Special issue: Proc. of the Workshop on Distributed Multimedia Systems, held in Graz, Austria, Nov. 1994), pp. 206-220, Springer, April 1995.

[8] HTTP, HyperText Transfer Protocol, http://www.w3.org/Protocols/

[9] HTTPS, HTTP Over TLS, http://www.ietf.org/internet-drafts/draft-ietf-tls-https-04.txt

[10] The Secure HyperText Transfer Protocol, http://www.ietf.org/rfc/rfc2660.txt

[11] R.L.Rivest,A.Shamir,L.M.Adleman: “A method for obtainingdigital signatures and public-key cryptosystems“, Communications of the ACM,21, (1978), 120-126

[12] Public Key Infrastructure (X.509)(pkix); http://www.ietf.org/html.charters/pkix-charter.html

[13] Internet X.509 Public Key Infrastructure Data Validation and Certification Server Protocols <draft-ietf-pkix-dcs-03.txt, http://www.ietf.org/internet-drafts/draft-ietf-pkix-dcs-03.txt

[14] Internet X.509 Public Key Infrastructure Certificate and CRL Profile, http://www.ietf.org/rfc/rfc2459.txt

[15] Summary of Tools for the JavaTM 2 Platform Security, http://java.sun.com/products/jdk/1.3/docs/guide/security/SecurityToolsSummary.html

[16]Java security API, http://java.sun.com/products/jdk/1.3/docs/api/java/security/package-summary.html

[17] Package java.util.jar, http://java.sun.com/products/jdk/1.3/docs/api/java/util/jar/package-summary.html

[18] Manifest and Signature Specification, http://java.sun.com/products/jdk/1.3/docs/guide/jar/manifest.html

[19] Info-ZIP Application Note 970311, ftp://ftp.uu.net/pub/archiving/zip/doc/appnote-970311-iz.zip

[20] WinZip Home Page, http://www.winzip.com/

[21] Jarsigner (Windows) , http://java.sun.com/products/jdk/1.3/docs/tooldocs/win32/jarsigner.html

[22] The MD5 Message-Digest Algorithm, http://info.broker.isi.edu/in-notes/rfc/files/rfc1321.txt

[23] SECURE HASH STANDARD, http://www.itl.nist.gov/fipspubs/fip180-1.htm

[24] PKCS #7: Cryptographic Message Syntax Version 1.5, http://www.ietf.org/rfc/rfc2314.txt

[25] The Java(TM) 2 SDK, Standard Edition, v 1.3 Beta Release, http://java.sun.com/products/jdk/1.3/

[26] XML-Signatur , http://www.w3.org/Signature/

[27] SDML - Signed Document Markup Language, http://www.w3.org/TR/NOTE-SDML

[28] SSL Proxy, http://www.obdev.at/Products/sslproxy.html

[29] Internet X.509 Public Key Infrastructure Time Stamp Protocol (TSP) <draft-ietf-pkix-time-stamp-04.txt, http://www.ietf.org/internet-drafts/draft-ietf-pkix-time-stamp-04.txt

[30] US Patent no. :5,022,080 Electronic Notary; Filing date: April 16, 1989; Inventors: Robert T. Durst, Kevin D. Hunter http://www.clir.org/pubs/reports/graham/intpres.html

[31] Transparent Content Negotiation in HTTP; http://www.ietf.org/rfc/rfc2295.txt

[32] Peter S. Graham: "Intellectual Preservation: Electronic Preservation of the Third Kind", March 94, available online: http://www.clir.org/pubs/reports/graham/intpres.html