Leveraging Internet-98 Technology for Computer Healthcare Networks: to its Limits and its Limitations

Gavin J. Brelstaff

CRS4, Center for Advanced Studies Research and Development in Sardinia
Via Nazario Sauro 10, 09123 Cagliari, Italy.
Email: gjb@crs4.it.

As submitted to:
IEEE Transactions on Information Technology in Biomedicine
Special Issue on
EMERGING HEALTH TELEMATICS APPLICATIONS IN EUROPE

April 1998

Note: all figure and their captions are in-line.
References are cited by name in this draft.

Acknowledgement:
This work was supported by the Region of Sardinia, and Hewlett Packard Italia.
The author is grateful for discussions with G. Zanetti and help from all members of the BMA at CRS4.

Leveraging Internet-98 Technology for Computer Healthcare Networks: to its Limits and its Limitations

Abstract

To what extent can current Internet technology be leveraged to fulfill the vision of the electronic patient record (EPR) as a multimedia object and the healthcare information system as a secure distributed computing network? We explore provision of reliable, secure, intuitive, and inexpensive medical Intranets through simple scripting and configuration – avoiding the need for large programming teams. By prototyping the EPR as a secure newsgroup we demonstrate the feasibility of a basic workflow system that: preserves a signed-paper style visibility of patient data at all times; enriches presentation with multimedia online image exam viewing and user controlled animation; whilst protecting confidential patient data via encrypted data transmission, digital signatures, and authenticated user-access control. In the process several limitation of this technology are uncovered.

Introduction

A controversial aspect of the rise of the Internet and its technology has been the potential to use essentially *free* software to enable information exchange between remotely cooperating professionals. Surely it has to be unsafe to trust such software particularly in the healthcare sector where patients' lives can be at stake. However, the traditional proprietary client/server systems can be so expensive to commission whilst still remaining fairly low-tech [IOM 1996], or lacking clinical functionality [Keen 1998] it may be worth considering how Internet technology might actually be leveraged. To succeed it needs to satisfy stringent security concerns [Anderson 96, Clayton 1998] and integrate visualization of medical image exams. We share the vision of the electronic patient record (EPR) as a multimedia object and the healthcare information system as a secure distributed computing network [Sacoor 1997].

A defining feature of the Internet is that it runs on free software (e.g. the Apache web server) without which most of the Web would not exist. Computer software has always has always been crafted by talented people outside of commercial imperatives (witness: Perl, GNU, and Linux). Public domain software is now recognized of having a pedigree of its own and has often been more rigorously tested than anything produced by commercial software houses. In place of the usual "babble" of incompatible proprietary formats (e.g. MS Word 6 & 7) open standards are adopted. These are often the fruits of independent bodies (e.g. HTML; HTTP; CGI) such as the W3C consortium guided by long term interests for forward and backward compatibility. The public domain also delivered the first leading Internet browser Mosaic. The Internet browser is a remarkable variation of the trend – for three years it has been developed by commercial teams (first at Netscape then also at Microsoft) to be "given away" as loss-leaders to the public. The browser war is about establishing control over the future direction of the Internet and thus its auxiliary market – not selling browsers. The ensuing competitive development has carried the original basic browser through an accelerated evolution so that now both Netscape and Microsoft's current version 4 browsers deliver a remarkably sophisticated set of similar features at no cost. This evolution has spawned two new open standards that may stimulate serious applications in such domains as medical information systems, namely: the client-side JavaScript (CSJS) scripting language and the Secure Socket Layer (SSL) protocol.

The main auxiliary markets is that of the *Intranet*: a private internet built inside an institution or enterprises that can provide an inexpensive, alternative to traditional distributed network applications (client server database access). Families of servers have been developed Intranets expanding the functionality of existing public domain servers and allowing a tighter degree of integration and management. At issue here is the evaluation of this technology's potential for direct transplantation into distributed healthcare communications. We need to distinguish whether it is capable of providing more than a mere powerful but semi-reliable toy for *surfing the web*. Can it be integrated to render medical intranets that do useful tasks reliably, securely, efficiently and inexpensively? How much of this can be achieved:

- Using simple configuration and scripting;
- Without having to resource and maintain an army of programmers;
- Without entirely re-equipping the existing computer-base of an institution;
- Without being subverted into proprietary (lock-in) technology of the browser providers?

We may appear to be tying our hands unnecessarily behind our back by refusing to program, but whatever can be achieved in this manner should be particularly applicable in less-favored regions where IT professional "hands" are often a scarce resource, and where *next-generation Internet* may not arrive for some time to come. This approach complements a current trend towards the deployment of cross-platform, object oriented, distributed computing such as the Telemed project at Los Alamos where having demonstrated some capabilities of previous browsers to convey *virtual patient records* [TeleMed 96] – work has now moved on to implementations via Java programming and CORBA interfacing. Although that approach can have benefits it does tend to impose delays while clients wait to download the Java *bytecode* necessary to run the applications. This is particularly frustrating if an installed browser already contains identical functionality (e.g. form-based requests, or basic image viewing). Moreover, alternatives to Java/CORBA middleware can require a lot of code to gain similar cross platform status (e.g. the HANSA project [Blobel and Holena 1997] took two million lines of code to define a Distributed Healthcare Environment API that runs on both Unix and Win NT). A clear case can be made for a component-based system architecture that allows intelligent use of small subsets of the whole information system without needing access to all of its parts [Rosenberg 1997]. We intend to harness existing browser components (e.g. readers, mailers, security modules, etc), whereas Java developers must re-implement these and may adopt the emerging component architectures of JavaBeans [Sun 1998] or CORBA-3 [OMG 1998]. Software components ideally should map on to system sub-units that, like black boxes, satisfy, whenever possible, discrete element of the system's functionality without needing general connectivity to the rest of it. Establishing key components early on leveraging browser parts, should have several benefits:

- The human computer interface retains an essential degree of familiarity and transparency. The user may already know and predict the behavior of the user interface components – and so may feel ready to confirm important decisions digitally.
- To achieve functionality beyond the capabilities of the browser a *roadmap* already exists to implement platform neutral, distributed components using Java/CORBA [Orfali and Harkey 1997].
- System design can be achieved incrementally by upgrading components. Work on any one component might proceed fairly independently – easing any development.

The goal of software *re-use* has driven a trend towards object-oriented cross-platform components. Whenever previously developed software can be re-used without going back to the program code system development ought to accelerate [Brooks 1995]. Healthcare institutes that successfully adopt the components within Internet browsers and Intranet servers would demonstrate exemplary re-use.

Extending the paper metaphor: Patient record as secure newsgroup

For text-based reporting, computer interfaces that mimic paper forms have proven intuitive and have remained popular ever since their early days on dumb-terminals connected to an SQL database (DB). The mundane interactive text field that can be filled-in and modified now constitutes part of the furniture of the human computer interface. The Internet has extended the paper-form metaphor into the realm of remotely accessible, platform-independent, multimedia presentations. The WMED system [Brelstaff and Loddo 1996] demonstrated this aspect by augmenting medical images with clinical text extracted from a relational DB. Its web server ran a CGI script written in Perl that linked to images, made SQL queries then formatted the replies into HTML. CGI – the Common Gateway Interface - remains the main workhorse for serving DB content across the web (despite the advent of superior programming technology: e.g. JDBC, WAI, Apache-ModPerl). Unfortunately however, most data entry methods drop the intuitive paper metaphor as soon as a form gets submitted. Transparency is immediately lost –medics neither know where the data has gone nor whether it was manipulated during the process to misrepresent them. With a paper form they would fill it in sign it and make a photocopy for their files before dispatching it. That corresponds much more to sending a *digitally signed email* than a CGI form. In fact, addressing the email to a *secure newsgroup* allows a digital photocopy to become rapidly available to all who need to consult or update it. A single news host could contain a newsgroup for each patient in a hospital department so as to implement a collection of EPRs. For each patient the corresponding newsgroup (see Fig. 1.) could contain a pre-specified set of *threads* that map on to each distinct section in a paper patient record (e.g. identity data: *generality*, clinical plan, diagnoses, vital signs, etc). Each of these threads would be initialized with a blank form appropriate to that section. As the patient progresses within the department medics would successively access and update the latest version of the form. Thus a temporal audit of the state of the form gets seamlessly compiled and becomes non-reputable whenever medics digitally sign their submission.

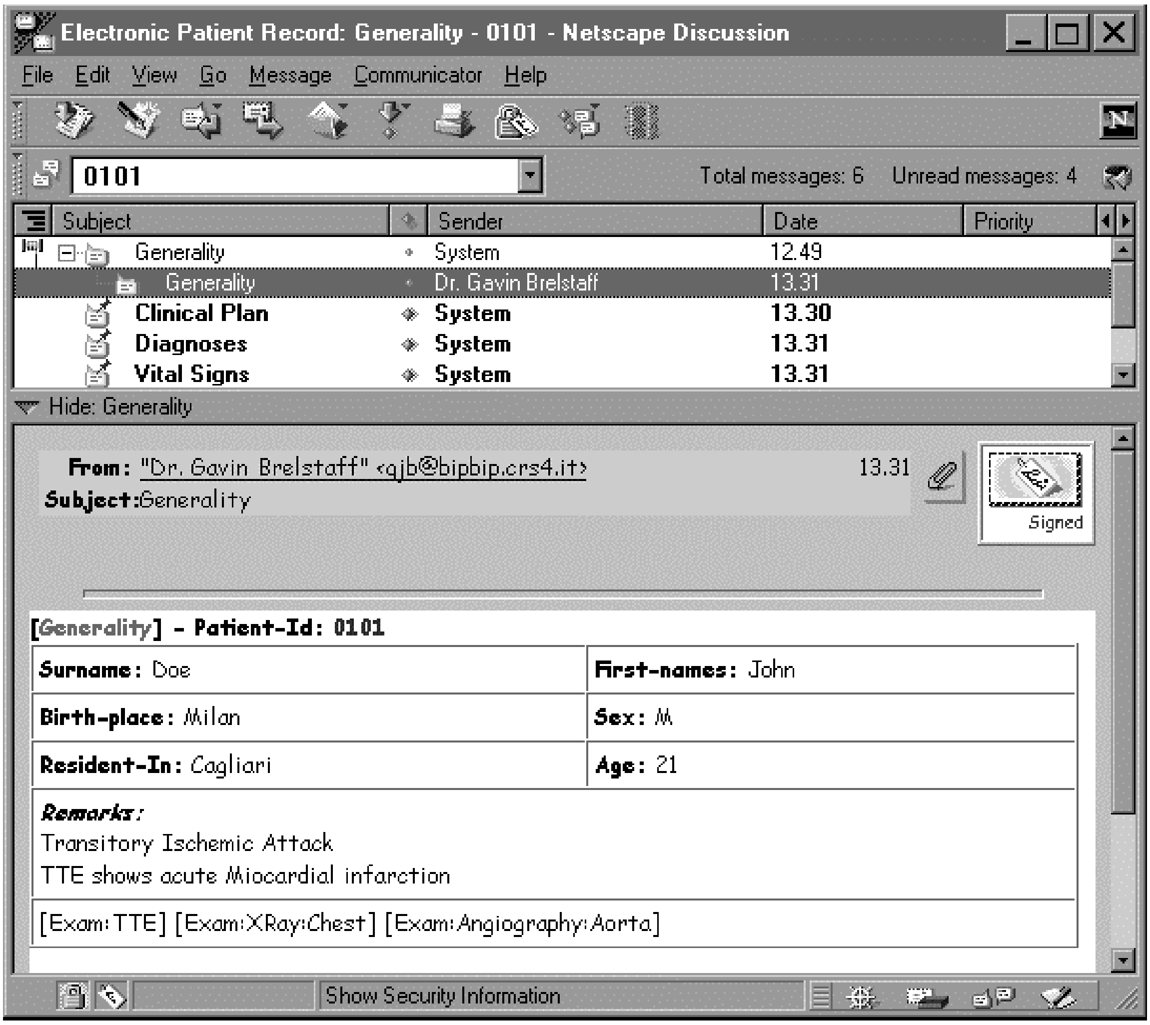

Figure 1: *Implementing an electronic patient record as a secure news group. Here patient 0101's generality is visible as an HTML page digitally signed by the medic who wrote it.*

This represents a primitive variety of a medical *workflow* system. A workflow system typically tracks the flux of digital documents around pre-specified paths in an enterprise (c.f. Lotus Notes). Ours could promote asynchronous communications between medics - which evidence suggests can often be more effective than direct means [Coiera and Tombs 1998]. For example, a radiologist may need to be alerted that a request has arrived for an examination. He may then schedule that request for a certain date and post a reply. After the exam takes place he would then follow-up with the submission of his report. This is feasible using the pre-structured newsgroup of successively refined forms as outlined above. Workflow more usually directs messages to personal mailboxes. There the radiologist would check his email for new requests for work. To maintain network-wide visibility of the work queue the email would need to reside on a central mail server.
A secure IMAP4 mail server (not POP3) might serve that need. IMAP4 and secure news protocols, in fact, share many similarities [Udell 1998]. However, an IMAP4 server is really designed for private (not group) reading – so imposing a structured grouping and threading with IMAP4 is less stable than with a newsgroup. Note, however, newsgroups do not implement transaction management (TM) so unlike commercial medical workflow systems (e.g. [Schmidt 1997]) it is difficult to prevent two medics simultaneously compiling the same form. Nevertheless, it is

interesting to illustrate what can be done without TM, while bypassing venerable formal message passing protocols such as EDIFACT [UN 1998] or HL7 [HL7 1996]. To do this, some technical details remain to be resolved:

- How can form data be mailed and still stay like paper?
- How can a mailed form be signed?
- How can patient confidentiality be maintained?
- How can medical images be furnished to the browser?
- What Intranet components are required?

These issues are addressed in turn in the remainder of this paper.

How can forms be mailed – and still stay like paper?

Sending an HTML form by email is more tricky than posting its data to a CGI program. This is because the data filling in the form's blanks is transient in nature as it does not constitute part of the page's HTML description. Naively filling in a form then using the browser's *Send Page* facility to launch the mail program with that page pre-attached just results in a blank form getting mailed. Somehow, the form's data needs to be fused into its HTML before the mailer gets launched. Fortunately, this can be achieved using CSJS. CCJS allows you to access the data in the form and then overwrite the HTML on the page (a primitive but stable form of dynamic-HTML) to reflect the data in a permanent manner. Thus a frozen version of the form can be written with an identical layout to the original – but with no longer any means of interactively modifying data. Syntactic care is required to correctly transpose each of HTML's various form elements (text, textarea, checkbox, select, button, etc) into its corresponding fixed text version. The process of freezing the form can be initiated in response to the click of the form's *Commit* button. This button is not present in the frozen version, but is transposed into a means of re-activating the form. Reactivation may be applied immediately by the medic to make a correction before mailing, or else it may be made by anyone responding to the message in the newsgroup – see Fig. 2.

To achieve a robust reactivation mechanism in the frozen form we inserted a hypertext link labeled *Activate Form*. This link when clicked calls the HTML page of the original blank form with a *search string* attached at the end of the URL containing all the data that was frozen in the form. Thus the original blank form has to contain additional CSJS to enable it to parse the data in the search string (like a CGI program does) and then to enter that data as values in the appropriate form fields. In effect, we are using a re-entrant script that has two states:

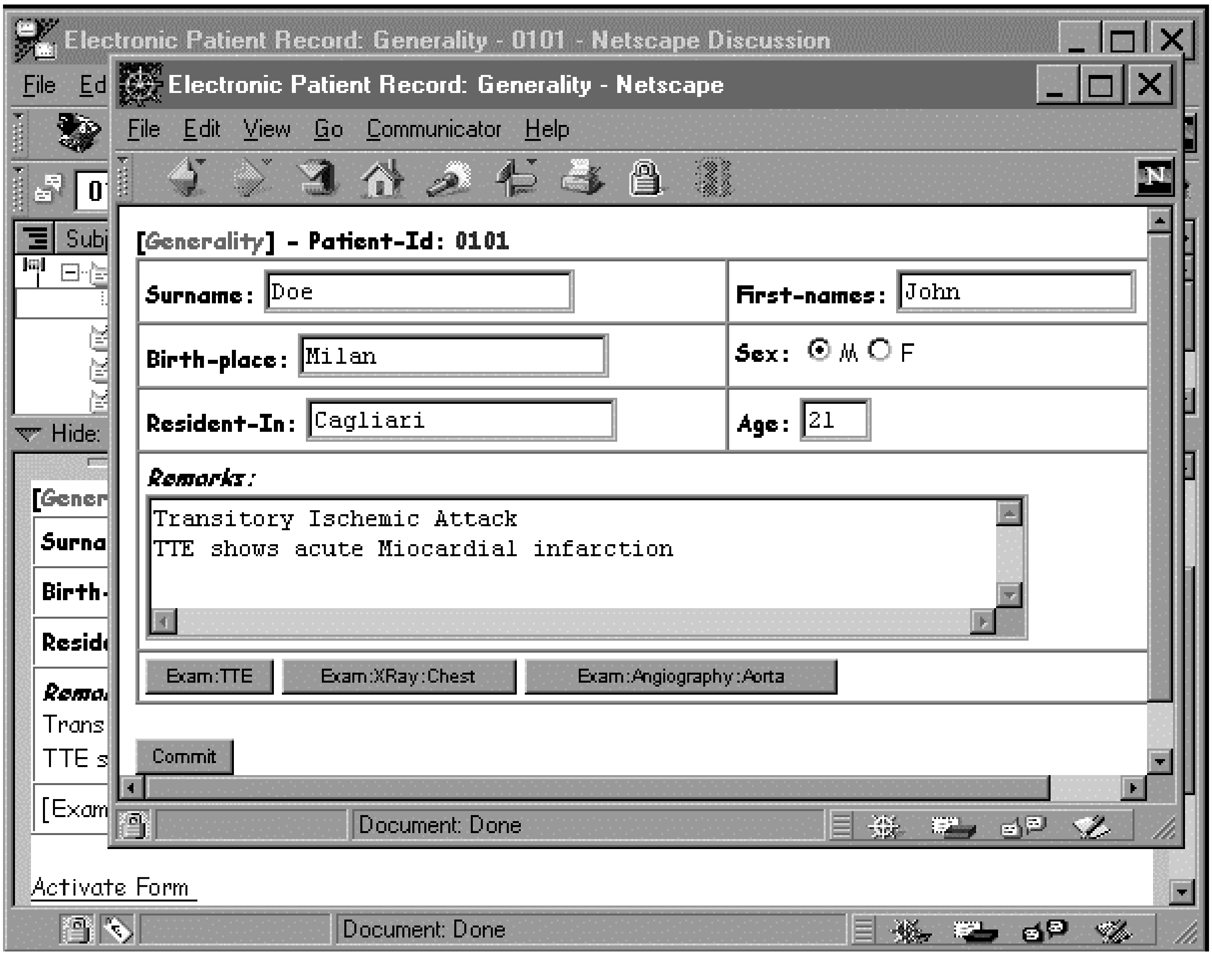

Figure 2: *The form previously submitted to patient 0101's generality section is stored in the newsgroup as a frozen HTML page, but it can be re-activated by anyone authorized to read it by simply clicking on the Activate Form link. The resultant form (raised) can then be resubmitted once updated by clicking on the Commit button. Furthermore, imaging exams can be viewed as pop-up windows (e.g. Fig. 4) by clicking the named buttons.*

- Frozen when in stored in the newsgroup, and visible in the news reader.
- Active once launched into the browser – by clicking *Activate Form* in the news reader.
- Frozen (again) in the browser – having clicked on the *Commit* button.

In the latter case the page is ready to e-mail to the newsgroup from where a subsequent work cycle can begin. It would be ill-advised to leave the process of addressing the email to the medic – any mistake may send confidential data to a random recipient! For this reason, the CSJS function that handles the clicking of the *Commit* button carries out a second task. It launches the mailer with the address, and subject of the message pre-specified - corresponding to the patient (newsgroup) and the section (news-thread).

This is achieved by co-calling a link specified according to the *mailto:* protocol. The mechanism by which the page gets attached is somewhat dependent on the browser type and version. The ideal browser appears to be Netscape 4 running on Unix or a Mac where the entire page can be prewritten as HTML directly into the message of the mailer merely by specifying the *&body=myHTML* field of the *mailto:* protocol. Testing for Internet Explorer 4 has been confounded by mailto configuration problems. However, Netscape 3 and 4 on Windows require the user to learn three or four fairly precise mouse clicks that navigate the attachment dialog and drop in the URL listed under the *Page Info* window.

How can a mailed form be signed?

Public key (PK) cryptography permits the signing of digital messages. The infrastructure necessary to sign and verify messages is built-in to Netscape 4. This infrastructure comprises both the handling of digital certificates containing the necessary keys and the module required to implement the S/MIME protocol [RSA 1996]. As Fig. 3 shows, each participant must be assigned a private (secret to them) and public key. Something encrypted with the private key can only be decrypted with the public key and vice-versa. Signing a message involves deriving a message digest (MD) from the private key and the message contents. Anyone possessing the public key can reverse the cryptographic process to verify the message. The private key acts as secret token that vouches for the individual's identity and the MD indicates that the message has not suffered tampering. (Note, that since Netscape 4.04 it has also become possible to sign text and CGI forms - but not HTML - using the same certificates and keys.)

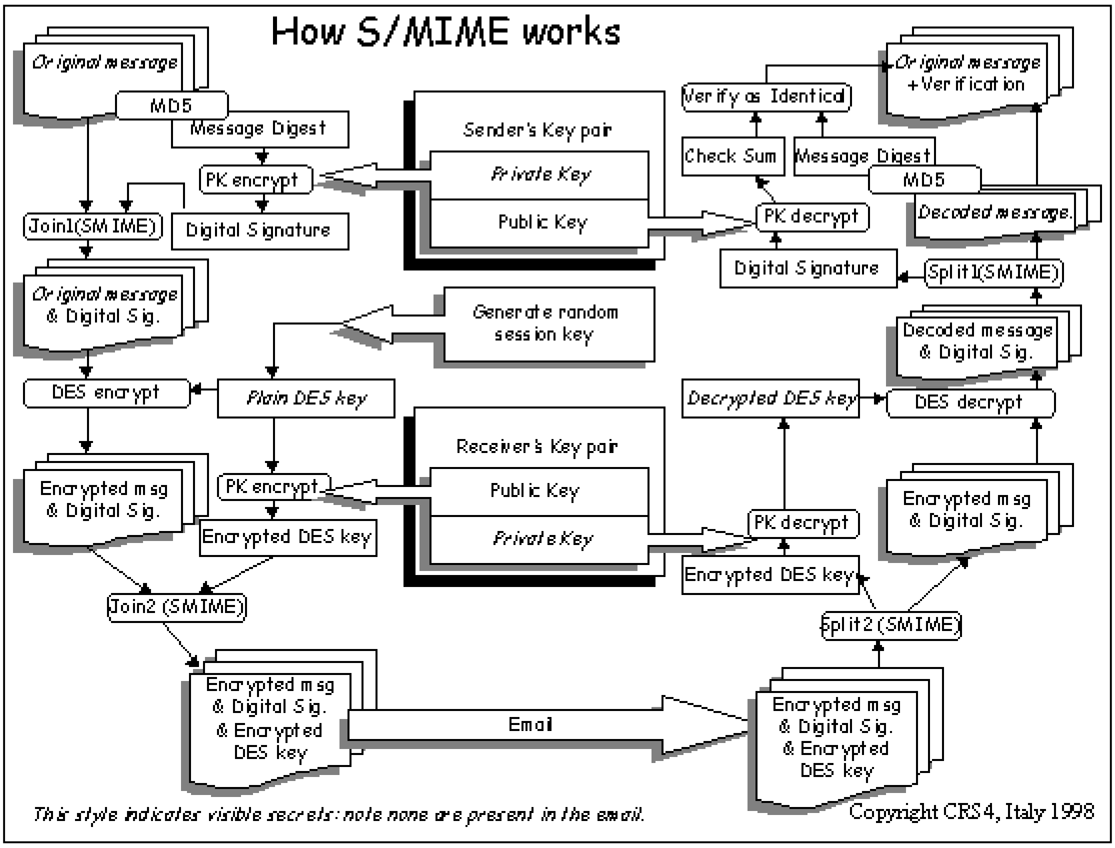

Figure 3: *An illustration of how the S/MIME secure email protocol allows emails to be digitally signed, and also encrypted when appropriate.*

How can patient confidentiality be maintained?

To prevent breaches of patient confidentiality all data ought to be cryptographically protected where ever it travels:

- On the newsgroup server
- In transmission between client and server
- On the client computer.

Ideally, *end-to-end encryption* would solve this entire problem: the data would be encrypted by the mailer of the submitting medic and would remain so up to the point the next medic's news reader decrypts it and makes it temporarily visible. As Fig. 3 indicates S/MIME can also provide end-to-end encryption, and it is available in Netscape 4 when the right certificates are in place. Unfortunately it is currently unsuited to communal group readership because each encryption is destined for only one recipient – the one with the matching private key. Although it might be theoretically possible to distribute the same private key to all participants of the newsgroup, that would currently effectively render them as the same digital identity and so would make nonsense of any digital signatures. This will remain the case until the arrival of separate signing and encryption keys within browsers [Elgamal and Cotter 1998].

Until then, a partial solution is to implement a secure newsgroup that encrypts all data in transmission between client and the newsgroup and vice-versa. Netscape's Collabra News Server achieves this using the standard SSL protocol. Previously, SSL has been used to ensure a secure web server [Hirsch 1997] such as the WMED system. For news or mail servers the technology is identical: a temporary secret session key (not PK) is transmitted to the client by a preliminary PK negotiation that establishes then authenticates the identity of the server and also the client when required. That session key is then used to encrypt and decrypt (e.g. via DES) the data at client and server (at full strength in Europe via Fortify [McKay 98]). In this case, physical access to the computer containing the newsgroup should be restricted as the data stored there is unencrypted.

Similarly measures need to be in place to prevent the medic leaving confidential files on the client computer. In general, client-side security must make a compromise that allows the medic to export some confidential data to disk or printer (i.e. trust them) whilst at the same time supporting then from unintentionally disclosing information to subsequent users of their computer (i.e. retaining their trust). For the latter case, some partial solutions do currently exist:

- Time-outs can be specified – so that an HTML form (containing sensitive information) that has not been submitted after, say, three minutes is replaced on the browser window by a blank page.
- No-cache directives can tell the browser not to cache an HTML form for future access. Nevertheless, such directives cannot prevent CSJS from writing to the browser's memory-cache (as our freezing a form does). Reaccessing an old copy of a form through use of the *back button* might be avoided by implementing extra CSJS mechanisms.
- Browser configuration could prevent a history list (of the pages accessed by the medic) being left for others to read. This configuration is best done using a browser customized so that user cannot manually reverse it [Netscape 1998].
- Closing the browser after use forces the next user to adopt their own personal digital id – not that of the previous medic. This policy may prove difficult to maintain – as medics may have to leave the browser at a moment's notice during clinical emergencies.

Thus, client-side security remains somewhat patchy in today's browser-based systems.

Getting medics to trust a system also depends on the veracity of the software. Our CSJS script take care not to contravene any browser sandbox security model so we have no need to incur the bureaucratic confusion involved digitally signing JavaScripts. It is enough to serve scripts from a certificated SSL server so that the medic always has resort to check the originators identity if ever in doubt.

How can medical images be furnished to the browser?

Images that can be included on an HTML pages are generally of inferior quality than those which radiologist see on their clinical workstations. The JPEG and GIF formats are restricted to 8-bit pixels and so do not support extreme contrast stretching that radiologist might like to apply to see the particular detail that they are looking for. Nevertheless, for some purposes they have been acceptable [Johnson et al 1998].

Problems arise when you try to provide full radiological image quality over the web. First, you must provide the software that allows this imagery to be viewed and manipulated (e.g. OSIRIS [Ligier 94]) as a *helper application*, or else a *plugIn* or Java applet [Bayo 1997] of similar functionality. Second, you must contend with larger data volumes imposing longer download times: Either you lease a fast line or you can consider compressing the image data. Image compression is still rather suspected by radiologists so we aim to adopt a pragmatic approach that gives an simple set of choices to the medic that determine how the 12- or 16- bit data on the server should be *contrast-ranged* into the available bit range of a GIF image. Animation control can also be provided for a sequence of images (e.g. TTE). For example, the medic might prefer to first view, say, every fifth frame in the sequence (rapidly) before choosing to download the rest. The server does the work of the ranging from DICOM to GIF format in organized collections of frames. For this we are exploring a CGI interface into the CTN public domain DICOM PACS software [Moore 1994]. The role of CSJS on the browser is to mediate the request of ranging and sampling parameters and provide controlled animation. The medic may stop, restart, reverse, step-through and alter the animation so this surpasses GIF-89 technology. In practice, the smooth running of the animation requires the whole selection of frames to be downloaded into the browser's memory-cache. Only then is execution not interrupted by re-accessing data from the image server. Fig. 4 provides a snapshot of our cross-browser implementation.

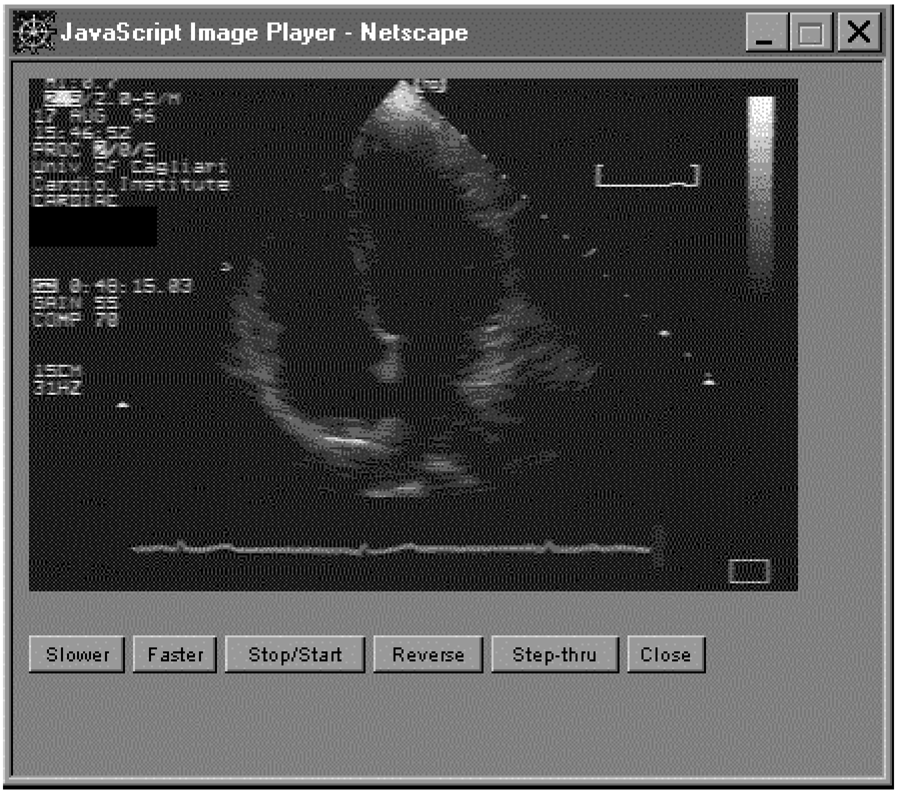

Figure 4: *A medical image viewer with fairly comprehensive animation control can be implemented using client-side JavaScript. Contrast ranging however must be done on the image server.*

Achieving other means of image compression (e.g. fractals or PNG) are not feasible in CSJS because it has no methods to process image pixels. Furthermore, serving images beyond the confines of a CGI interface appears suited to a CORBA PACS interface [Loddo 1997].

What Intranet components are required?

Our prototype EPR newsgroup has been implemented as a small Intranet illustrated in Fig. 5 the main operational components are:

1. A secure news server (Netscape Collabra) running SSL, and restricting access to only those users registered to use it. Access control is be either by verified PK certificate or by username/password.
2. A secure web server (Netscape Enterprise 3.5) running SSL using an identical access control procedure. It is used to serve: the re-activated CSJS-enabled forms, images via a CGI/CTN cache interface, and other pages such as on-line help, etc.
3. A directory server (Netscape DS 3.5) that provides a centralized user database storing all the information that the other servers need to control access and verify digital ids. It is also available to the browser (Netscape 4) to find the email addresses and PK certificates of the other users of the system.

As Fig. 5 shows administration (and installation) of all these servers is centrally controlled by an additional web server (SuiteSpot). Furthermore, we were able to manufacture all users certificates under our own certificate authority using a dedicated certificate server (Netscape 1.01).
Two aspects dictated our choice of Netscape servers: they are available for Unix not just WinNT and they are designed with extreme scalability in mind. Although, there was minimal technical support, we were able to evaluate the full products at no cost.
Note from Fig. 5, *no* relational DB is needed – the newsgroup itself constitutes the data archive. It may be useful, however, at least during a system migration phase or for other views such as data warehousing, to mirror each newsgroup update to an existing legacy database. This might be achieved simply by spawning an additional response to the *Commit* action that posts to the DB via a CGI program.

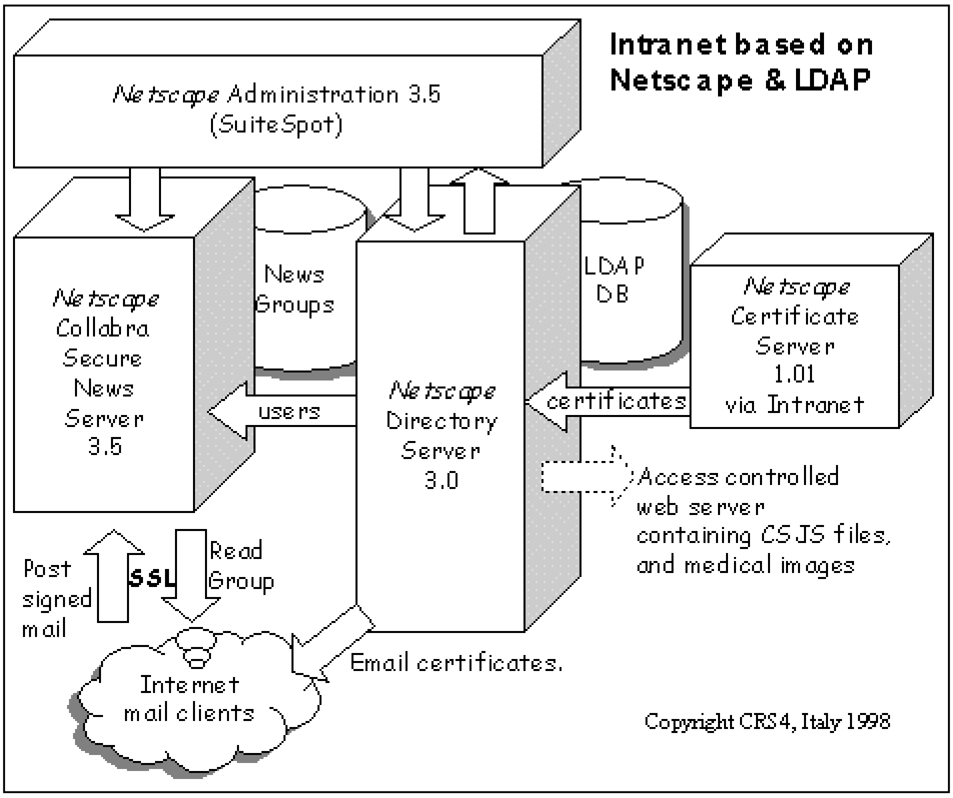

Figure 5: *An integrated infrastructure for digital identity such as that provided the LDAP directory service in Netscape family of servers can unify access control and digital signature verification across the news server, the browser, as well a web server used to serve images or CS JavaScript enabled forms.*

Conclusions

By prototyping the EPR as a secure newsgroup we have attempted to take current browser scripting to its limit. In the process we have been able to demonstrate a surprising degree of feasibility including:

- The basics of a workflow system that preserves a paper-style visibility of patient data at all times.
- Multimedia EPR presentation enriched by online image exam viewing with controlled animation.
- Protection of confidential patient data via encrypted data transmission, centrally administered user-access control, and attributable data

Several limitations have been uncovered including:

- The current lack of end-to-end encryption for group reading.
- The lack of built-in transaction management of the kind a relational DB traditionally provides.
- Any image enhancement still must be applied on the server as the browser cannot access individual image pixels.
- Similarly, browsers currently do not support vector graphics so ergonomic rendering of annotated overlays of image exams is practically impossible.

However, some of these limitations may be removed through the continuing activities of :
- the browser producers to extend functionality.
- researchers carrying out broader range projects on the EPR (e.g. PROREC [Redondo 1998]) and on trustworthy health telematics (e.g. TRUSTHEALTH [Klein 1998]).
- the W3C to steward new open standards.
- medical informatics bodies working towards refined standardization e.g.: CEN/TC/251, ACR/NEMA, and CORBAmed.

References

- R.J. Anderson, "*Security in clinical information systems*", London: BMA, 1996.

- J.F.Bayo, R.Gomez, X.Catusus, O.Barbero, M.Feron, M.Sentis, E.Bellon "DICOM Java viewer for a using WWW Internet Technology and DICOM 3.0 standard", Proceedings of EuroPACS, Tipografia Editrice Pisana, Pisa, C.Bartolzzi & D.Caramella (Eds), Italy, 133-136, 1997.

- N. Blobel, M. Holena, "Comparison, evaluation, and possible harmonisation of the HL7, DHE and CORBA middleware", *New Technologies in Hospital Information Systems*, J. Dudeck et al (Eds), IOS Press, 40-47, 1997.

- G. Brelstaff , S. Loddo "WMED – An experimental electronic patient record system based on the WWW." CRS4 Internal Report: http://www.crs4.it/ ~gjb/BMA, 1996.

- F.P. Brooks, "The Mythical Man Month: Essays on Software Engineering", Addison Wesley, 1995.

- P.D. Clayton. et al, "For the Record", Washington: National Academy Press, 1998.

- E. Coiera, V. Tombs, "Communicating behaviours in hospital setting: an observational study", *British Medical Journal*, No. 7132, Vol. 316, 28 Feb 1998.

- T.Elgamal, S.Cotter, "Netscape security: open-standards solutions for the enterprise", http://developer.netscape.com/docs/manuals/security/scwp/index.htm, 1998.

- F.J. Hirsch, "Introducing SSL and Certificates using SSLeay", *World Wide Web Journal*, Summer 1997.

- HL7 Technical Steering Committee Retreat: *HL7 Version 3*, Health Level Seven Inc, 1996.

- IOM, "*Telemedicine: a guide to assessing telecommunications in health care*", Institute of Medicine, National Academy Press, 1996.

- D.S. Johnson, R.P. Goel, P. Birtwistle, P. Hirst, "Transferring medical images on the world wide web for emergency clinical management: a case report", *British Medical Journal*, No 7136, Vol 316, 28 Mar 1998.

- G. Klein, "TRUSTHEALTH: Trustworthy health telematics", EU-Project:HC1051, http://www.ehto.be/projects/trusthealth/ 1998.

- J. Keen, "Rethinking the NHS networking", *British Medical Journal*, No 7140, Vol. 316, 25 April, 1291-1293, 1998.

- Y.Ligier, O.Ratib, M.Logean, C.Girard, "OSIRIS, a medical image manipulation system", *M.D. Computing Journal*, Vol. 11, No 4, 212-218. 1994.

- S. Loddo, G. Brelstaff , G. Zanetti, "A distributed heterogeneous image server", Proceedings of EuroPACS, Tipografia Editrice Pisana, Pisa, C.Bartolzzi & D.Caramella (Eds), Italy, 199-1202, 1997.

- F. McKay, "Fortify for Netscape", public-domain software, http://www.fortify.net 1998.

- S.M. Moore, S.A. Hoffman, D.E. Beecher, "DICOM Shareware: A Public Implementation of the DICOM standard", *Medical Imaging*, 1994.

- Netscape, " Mission Control Desktop", http://home.netscape.com/ Netscape Comm. Corp.1998.

- OMG, "*CORBA Architecture and Specification*", Object Management Group,1998.

- R.Orfali, D. Harkey, "Client Server Programming with JAVA and CORBA", John Wiley, 1997.

- J.R. Redondo, "PROREC: Promotion strategy for the European electronic healthcare record", EU-Project:HC1110, http://www.ehto.be/projects/prorec/ 1998.

- J. Rosenberg, "JavaX: an approachable examination of Java, JavaBeans, JavaScript and all related technologies", http://developer.netscape.com/docs/wpapers/javax/javax.html, 1997.

- RSA, "S/MIME Message specification: PKCS security services for MIME S/MIME Editor", RSA Labs, ftp://ftp.rsa.com/pub/S-MIME/smimemsg.txt, Feb. 1996

- D.Sacoor, "Integration of images for multimodal medical applications" Report of Workshop organized by the European Commission DG XIII, Brussels, March 1997.

- J. Schmidt, K. Meetz, Th. Wendler, "The application of workflow management systems in Radiology", Proceedings of EuroPACS, Tipografia Editrice Pisana, Pisa, C.Bartolzzi & D.Caramella (Eds), Italy, 257-260, 1997.

- Sun Microsystems, "The JavaBeans Component Architecture", http://java.sun.com, 1998.

- TeleMed, “The Virtual Patient Record”, http://www.acl.lanl.gov/TeleMed 1996.
- J. Udell, “Exploring the SSL modes of Netscape’s Messaging and Collabra servers”, Web Project: Securing Mail and News, April, *Byte*, http://www.byte.com. April, 1998.
- UN, “UN/EDIFACT standards”, http://www.premenos.com/unedifact , 1998.