Signatures: an Interface between Law and Technology

Ben Laurie
ben@algroup.co.uk
Nicholas Bohm
nbohm@ernest.net

January 30, 2003

1 Introduction

The aim of this paper is to explore the boundaries between the technical and legal worlds of cryptography, with particular reference to signatures, contracts and their meaning. The relevant legal context is that of England and Wales: others may have quite different features, even as close to home as Scotland.

2 Definitions of Terms

In the worlds of both information security and law, words have much stricter definitions than they do in other contexts. Cryptographic terminology is usually highly specific and frequently contradictory relative to "real world" definitions of the same words. Conflicts often arise because a techy's definition of a term may be different from a lawyer's. Usually when this happens, there's no clear signal between the two individuals that a mismatch of terminology has occurred.

2.1 Cryptography

Cryptography is traditionally thought of as the science of secret messages. In the modern technical world, secret message are certainly still included within the realm of cryptography, but it has also widened to cover other aspects of security authentication.

Public key cryptography (see 2.5), first revealed to the world in the 1970s[3, 4], permits the electronic equivalent of a signature to be constructed (see 2.6): in many ways electronic signatures are actually superior to their paper analogues, but in some they are less satisfactory. It is this mix of virtue and vice that lies at the heart of much confusion about the technical and legal meaning of electronic signatures.

2.2 Electronic Object

This is a term we will use frequently throughout this document, though it is by no means a "standard" technical term. Unfortuntely there really isn't a standard term for what we mean, because it is a concept so fundamental in the technical world that it is rarely explicitly named.

By an electronic object we mean a document, a program, a piece of music or any other similar thing. The essence of it is a thing that a computer can store which can be transferred from one machine to another.

Technical types will use words like "file" or "collection of bytes" for what we are calling an electronic object. It is important to understand that what you may think of as being the same "document" can easily be different "electronic objects" - for example, a word processing document can be saved as a file on disk. It can often be saved in mutliple formats - saving an identical document in two different formats results in two different objects, even though they probably will look exactly the same if you open them both in your word processor. Depending

on the word processing software, it may well be that saving the same document twice in the *same* format results in different objects!

Conversely, if you copy an electronic object, the copy is the same object - it consists of an identical sequence of bits (zeroes and ones), and if an electronic signature is generated (see 2.6) for the copy, it will be the same as for the original. It is important to understand that for electronic objects one copy is indistinguishable from another (or from the "original").

2.3 Secure Hash Functions

These are also known as one-way hash functions. Hash functions in general are also known as digests.

A hash function is simply a gadget that can take an electronic object and produce from it a number which is dependent only on the object. If the object is altered, so is the result of the hash function. If the object is identical, then the result of the hash will be identical.

A secure hash function is a hash function with some special properties. The most important of these from our point of view is that the output of the hash function cannot be forged - in the sense that given some particular output, it is computationally infeasible to construct from scratch an object that produces that output. As a result of this property, secure hash functions can be used to generate signatures (see 2.6) in various ways.

Only two secure hash functions are in wide use: MD5 and SHA-1. MD5 is generally not preferred these days, because there have been some successful attacks on parts of its innards. An MD5 hash is always 16 bytes long and an SHA-1 hash is always 20 bytes.

The length of a secure hash, as well as its cryptographic properties, is important. The reason is that the property of being unable to produce a different second document that has the same hash as the first is reliant on the size of a hash - if it is too short, once can simply keep making minor changes to the second document (e.g. changing spacing or unimportant words or phrases) until some variant of it produces the required hash.

2.4 Birthday Attack

It is important to note that being able to produce two documents with the same hash can be useful to an attacker, and that this is substantially easier then producing a document with a particular hash.

The method for doing this is known as a "birthday attack" - so-called because of the fact that if you ask a relatively small group of people (anything over about 25 in fact), there is a reasonably good chance that two of them will have the same birthday, a fact which most people find surprising, because intuition suggests that a much bigger group would be needed to produce this result. This is generally true for any random list chosen from a finite set.

In order to apply this attack to the hashes of objects, one takes the two objects and makes a series of inconsequential changes to both of them, hashing them

each time, to produce a list of hashes for each document. After a number of attempts (about equal to the square root of the number of possible hash values), there's a good chance that some version of the two objects which produce the same hash will have been constructed.

The purpose of this attack is typically to get the victim to sign one document, then use the signature as if they had signed the other. Clearly hash functions must produce sufficiently long output to resist this attack. Because this attack effectively reduces the difficulty of an attack to about half the number of bits of the hash, it means hash functions have to produce about twice as much output, to be safe, as you would otherwise think they did.

The widely used MD5 and SHA-1 hashes are both sufficiently long to be effectively immune to this attack.

2.5 Public Key Cryptography

Public key cryptography is one of the main technologies supporting electronic signatures and contracts.

The essence of public key cryptography is the public/private key pair (often simply known as "the key pair"). This is a pair of cryptographic keys that have the interesting property that anything encrypted with the public key can only be decrypted with the private key, and anything encrypted with the private key can only be decrypted with the public key.

The usual practice is to keep your private key secret and known only to yourself (for some definition of "yourself") and to allow the world at large to know your public key. It is vital to understand that it is infeasible to figure out what the private key is, given the public key[1].

2.6 Electronic Signatures

An electronic signature is, in general, an electronic object which, by cryptographic means, binds a particular electronic object to a particular private key. That is, the signature can only be produced by one particular electronic object in combination with one particular private key. Since it is supposed that the private key is only available to a particular person (or group of people), the signature object in effect guarantees that that person (or group of people) produced the signature for the particular object.

The standard way to produce an electronic signature is to take a secure hash (see 2.3) of the object to be signed and encrypt it with the appropriate private key. Since the encrypted version of the hash can only be produced by that private key, and the hash can only be produced by that particular object, and it can be verified (by decrypting the encrypted hash with the corresponding public key) that the hash was encrypted by that particular private key, the signature is evidence of some kind of intent (i.e. the intent by the owner(s) of the key to sign the object).

[1]Note that there is no theoretical reason that it should be possible to figure out the public key given the private key, either, but it so happens that it is generally possible to do so

It is worth noting that the use of the secure hash, rather than the document itself, as the thing to be encrypted is partly due to technical limitations of the known types of public key cryptography (in short, public keys can only encrypt things up to a certain size) and partly because public key cryptography is generally pretty slow, so encrypting small things like hashes rather than big things like contracts is generally favoured.

Legally, an electronic signature means

> so much of anything in electronic form as-
> (a) is incorporated into or otherwise logically associated with any electronic communication or electronic data; and
> (b) purports to be so incorporated or associated for the purpose of being used in establishing the authenticity of the communication or data, the integrity of the communication or data, or both.[1]

Electronic signatures as understood by techies fit this definition, but so do many other things (see 9).

2.7 Non-repudiation

"Non-repudiation", in its technical sense, is a property of a communications system such that the system attributes the sending of a message to a person if, but only if, he did in fact send it, and records a person as having received a message if, but only if, he did in fact receive it. If such systems exist at all, they are very rare.

Non-repudiability is often claimed to be a property of electronic signatures of the kind described above. This claim is unintelligible if "non-repudiation" is used in its correct technical sense, and in fact represents an attempt to confer a bogus technical respectability on the purely commercial assertion the the owners of private keys should be made responsible for their use, whoever in fact uses them.

Some systems propose that public keys should specify whether or not signatures verifiable by their use are repudiable or non-repudiable. This proposal is of very doubtful use - non-binding signatures are of little value, and if a document is intended not to be legally binding, it can achieve this result by incoporating a statement to that effect. This seems simpler, and fairer to unwary readers, then embedding the statement in the signature or the key, where it may not be noticed.

To lawyers, non-repudiation was not a technical legal term before techies gave it to them. Legally it refers to a rule which defines circumstances in which a person is treated for legal purposes as having sent a message, whether in fact he did or not, or is treated as having received a message, whether in fact he did or not. Its legal meaning is thus almost exactly the opposite of its technical meaning.

Such a rule may be imposed by law, as for example this rule:

> The person making the return to the Controller shall be presumed to be the person identified as such by any relevant feature of the electronic return system.[2]

It can also be imposed by contract, as for example this rule governing access to the National Land Information System provided by a firm of private contractors:

> You will be liable for all charges incurred through the use of your password.

2.8 Certificate

A certificate is an electronic object that associates a public key (and hence the corresponding private one) with some other information, usually identity information or permissions of some description. A certificate is signed (see 2.6), at least according to some infrastructure proposals, by a certificate authority (CA) (see 2.11), who usually attests in some way (at least in theory) to the link between the identity or permissions and the holder of the private key. To lawyers, a certificate can be on paper and need have nothing to do with keys at all.

The advantage of certificates is that it is possible to check them using the public key of the CA. This means that it is possible to introduce a new user into a system without the system itself having to know about it.

Unfortunately much of this advantage is removed by the need in many cases to check the status of the certificate whenever it is used (see 2.14).

The most common use of certificates currently is with HTTPS (see 2.16) to verify the identity of secure webservers. In this case the identity associated with the public key is the domain name of the webserver. Before the CA issues a certificate linking the domain name to a public key, it checks that the domain belongs to the person or entity (usually a company) that has requested the certificate. It also checks that it really is dealing with that entity and not some imposter. Errors in these checks have potentially serious consequences (for example, in January 2001, Verisign erroneously issued to an unauthorised outsider a code signing certificate bound to Microsoft's identity, which could have been used to inject arbitrary code into unsuspecting users' machines[5]).

Another, less common but increasingly popular, use of certificates is what are known as "client certificates" - these are issued to individuals as a proof of identity that is independent of any particular server or system, and, indeed, exemplify the supposed merits of certificates. It is, for example, now possible to complete VAT returns electronically, but to do so, one must have an identity certificate issued by a recognised CA.

2.9 PGP

"PGP" stands for "Pretty Good Privacy". PGP is a system for encrypting and signing objects, using public key cryptography, that does not rely on any kind of central authority - instead keys are verified by peer relationships - that is,

each key is signed by other PGP keys - in the end, any user's trust in a key should depend on their trust in keys they have personally verified. PGP has been standardised as OpenPGP[7].

PGP is most often used for encrypting and signing emails, usually between individuals or organisations, but is also often used as a part of a secure e-commerce infrastructure - typically to encrypt orders being sent from a webserver to the back office that will fulfill them.

2.10 X.509

X.509[8] is an ITU standard for digital certificates - originally designed for securing mail directories, it has been adopted for use in SSL (see 2.15).

In X.509 a public key is associated with a Distinguished Name (or DN), which is a ghastly mass of data supposedly defining the identity of the certificate holder in a hierarchical way. This fits with the model of X.500 (a mail directory standard that has largely failed to gain wide acceptance) but doesn't really fit well with how the world works.

2.11 Certificate Authority

A certificate authority (CA) is responsible for signing certificates. In order for there to be any value to this signature the CA must apply some kind of check to the certificate it is signing. In general public CAs do the check (or have Registration Authorities do it for them) but try hard to disclaim all responsibility for failure to do so correctly.

Private CAs (that is, those that are only used within an organisation) are more commonly associated with client certificates.

Note that one of the checks a CA should carry out is that the applicant is in possession of the private key corresponding to the public key in the certificate. This should be done by means of a signature on the certificate request, and not by the CA producing the private key on behalf of the applicant.

If a CA itself creates a key pair and issues it to the applicant, it can be sure that at that moment the applicant has the private key and that nobody else has access to it (except the CA). This is helpful to the CA as it can certify that the applicant is the only person in possession of the key. It is not helpful to the applicant, who should regard the CA's access to the key as a major security weakness. If the applicant insists on creating its key pair and presenting the public key to the CA, the CA cannot know that the applicant is the only person with access to the private key. But the CA cannot in any case know this for any time after the signing of the certificate, so it makes little difference.

2.12 Registration Authority

Sometimes a CA delegates the responsibility for checking identity (or other certified attributes). In this case the entity that does the checking of the certificates is known as the Registration Authority (RA).

2.13 Revocation

Revocation is the process of invalidating a certificate. This is normally done by signing some kind of instrument indicating which certificate has been revoked. This signature may be made by the same key as was used to sign the original certificate, or by a different one. The idea is that once a certificate has been revoked, no-one should rely on it. The problem with revocation is that the need to check it largely removes the advantage of certificates - rather than check whether the certificate has been revoked, why not get whatever was in the certificate from the appropriate authority at the time?

2.14 Certificate Revocation List

This is simply a list of revoked certificates. In the case of X.509 it would usually be a list of the DNs of the revoked certificates.

2.15 SSL/TLS

SSL[9] is the Secure Socket Layer. TLS[10] is Transport Layer Security. TLS and SSL are, in fact, very similar, the former having been defined by Netscape, and widely adopted, and the latter being the standardised version as defined by the Internet Engineering Task Force (IETF).

SSL and TLS usually use X.509 certificates to verify at least one end of the connection (usually the server), and are best known when used in HTTPS.

Note that unless one end of the connection is verified somehow, then the connection is susceptible to a "man in the middle" attack - where an eavesdropper sits between the two parties trying to communicate and pretends to each one to be the other, relaying all messages between them, but thus having access to the decrypted messages.

2.16 HTTPS

HTTPS is simply HTTP secured by running it over SSL. And HTTP is the Hypertext Transfer Protocol[11]. HTTPS is what you use whenever you go to a secure webserver. It has two functions - first to keep confidential data private (by encrypting the link it is transmitted over) and second to verify the identity of the server you are dealing with (or, more importantly, its owners).

2.17 Randomness and Entropy

Much cryptography requires random numbers to make it secure. For example, when choosing a public/private keypair, it is important to ensure they can't be guessed easily by an attacker. This is achieved by picking them randomly. Similarly, when an HTTPS session is established, a symmetric key is chosen at random to encrypt the session - again it is a good idea if that key cannot be guessed. The techy term for the "goodness" of random numbers is, essentially, entropy. Entropy is a measure of the "amount" of randomness available. It is a

good idea, in most cases, to have at least as much entropy available as the size of the random number you are picking.

2.18 Threat Model

It is rarely useful to define a cryptographic system without having some idea of what you are trying to protect against. This is known as the "threat model". There is no faster way to earn the scorn of a security practitioner than to propose some secure scheme without defining the threat model.

For example, the threat model commonly defined for the Web is that attackers will attempt to steal credit card information or other personal information. Two widely considered methods of doing this are by eavesdropping, or by masquerading as the webserver. SSL purports to solve both of these. Certainly it does solve the problem of eavesdropping, by encrypting the conversation with a key that can only be known to the two ends. But it is less clear that it solves the second - SSL verifies the identity of the server by checking that its domain name matches its certificate (and that the certificate was issued by a "trusted" CA, of course). But there are at least three problems with this check - first, if it fails, the user is issued with dire warnings, which are almost always ignored. Second, the CA, as discussed above, may not exercise all that much diligence in checking the credentials of the certificate holder, and may disclaim all liability for any mistakes. And third, and perhaps most seriously, the URL is not all that strong an indicator of the identity of the server's owner. For example, who owns www.micr0soft.com? Probably not Microsoft. If you write it as WWW.MICR0SOFT.COM, it's very hard to tell from WWW.MICROSOFT.COM, isn't it? Although the certificate does contain the identity of the owner (assuming the CA has done his job), almost no-one checks it - or checks revocation lists.

3 Are PGP Signatures legally binding?

To make sense of an answer to this question, it's best to start by thinking about signatures on paper.

Is your signature on a birthday card to your mother legally binding? Not usually, because there isn't usually anything in a card by which you can be bound, and even if there were (like a promise to pay her a visit if she promises to pay your bus fare), family transactions like that aren't normally legally binding.

Is your signature on a cheque to pay a bill legally binding if you write in pencil in your left hand (assuming you're right handed)? Yes: the bank may dishonour it if it doesn't recognise the signature, but the person you give it to can sue you on the cheque if he can show you signed it (for example, because he saw you sign). Note that he doesn't have to produce two knights and a bishop as witnesses, or fulfill any other formal requirement, although he might want to call expert evidence from a document examiner. It is enough for his evidence to be believed, on the balance of probabilities, in the face of any denial by you. This means that his evidence has to show that it is more probable than not that you did in fact sign the cheque.

If a crook forges your signature on a cheque well enough to get your bank to cash it, is that signature binding on you? No, if your denial is believed. It is up to the bank to prove that its debit to your account is justified if you challenge it, so it must prove on the balance of probabilities that the signature is yours. If it looks very like yours, you might be wise to call an expert witness to say that on close examination it's a forgery.

If you sign a document to witness the signature of someone else, are you legally bound by the document? You're bound only by what your signature says, according to the context. In this case your signature says you saw who signed, and usually nothing more. You might be called as a witness to back up that statement in court, but you haven't agreed to the rest of the document.

A signature on paper is just any mark you make as a signature, namely so as to show your adoption or approval of the content of the paper, according to what the paper says. The essential point is that the meaning of a signature depends not on any characteristic inherent in the signature, but on the content of what is signed.

The rules for electronic signatures are just the same. Your PGP signature, like your paper one, is good in law if you made it, but not otherwise. You can accept a different rule by contract, like the access terms quoted above; or a different rule might be laid down specially under statutory powers, like the VAT electronic filing rule quoted above. But there is no general rule that things are different for electronic signatures. A fuller treatment of these topics has recently been published by the Law Commission in formal advice to the UK Government[6].

4 Meaning of Signatures

If I PGP (or otherwise) sign a piece of software (e.g. an Apache distribution) have I opened myself to any kind of liability (for example, if it has a security flaw, or if it contains a trojan)?

A signature means what the signed object says it means. You should read what you're signing, even if you wrote it yourself. If you're not sure, make it plain: include a comment to say that the signature is intended to authenticate the source of the code, and does not add to or vary whatever warranties are given about its content in whatever document specifies that.

Note that in general it is impossible to directly read what you are signing electronically, since it is likely to be a word-processing document or spreadsheet. Even "plain text" cannot be read without the assistance of a text editor, though at least in that case you can be reasonably sure there are no hidden surprises (unlike some embarrassing cases in which word processing documents have retained previous versions of the text but not displayed them to the originator).

This is, of course, in sharp contrast to paper documents where it is abundantly clear what is being signed. Much current system design completely ignores this problem. An obvious attack on an unwary user is to subvert the software that displays the document to the user and thus persuade them to sign something completely different from what they think they are signing. Even the much touted panacea for all security ills, the smartcard, does absolutely nothing to solve this problem.

The wide use of insecure operating systems and software seems likely to cause problems in this area as wider use is made of electronic signatures.

5 A duty of care?

Are you under a legal duty to take care to prevent unauthorised access to your private signature key?

There is no UK statutory duty to be careful with your key. There are cases where you are bound by what has been done with your key whether you did it or not, either as a result of contract terms or specific statutory rules applying to particular cases, which we mention at 3. (This has nothing to do with a duty of care, which is a seductive but misconceived basis for analysing responsibility for the use of cryptographic keys in commercial contexts.)

It is completely open whether any corresponding general rule will be developed by the common law to cover all those who publish or provide a public key: at best, we think, the courts might be persuaded that some corresponding term must be implied as the basis on which a key is provided, at least in a commercial context; but no such term could be implied if it were expressly negated. This could be done by an express statement, for example the following: "I will accept that I am bound by what I sign with the private key corresponding to this public key; but although I am not aware that anyone other than me has access to that private key, and although I intend to try to ensure that nobody other than me has or obtains access to it, I do not accept any legal obligation to make any such efforts nor do I accept any legal liability for any consequence of unauthorised access to my private key, whether or not I was careless in any efforts to safeguard it; and therefore I do not warrant or undertake that I am or will be the only person with access to that key, and I will not accept any responsibility for anything signed with it which is not positively proved to have been signed by me or with my authority. The fact that this key verifies a signature should therefore not be taken by itself to prove that I made that signature."

6 Is "non-repudiation" a useful concept?

No. The inconsistencies between the notions of (1) a non-repudiation rule and (2) a non-repudiation system property are so severe that most discussions in which the term is used are more confusing than helpful.

It is particularly unhelpful to apply the term to a signature. If I want to sign a document but not to be bound by it, then there is a perfectly standard way of achieving this result: I state in the document that it is not intended to be legally binding. There is even standard lawyers' jargon for the purpose: mark the document "Subject to Contract".

If I want to subject myself to a regime in which anything verifiable by my public key is binding on me whether I signed it or not, then I can publish an explicit offer to that effect, open for acceptance by anyone who relies on a document whose signature is verified by my public key. The effectiveness of this legal technique has been recognised by the courts since the nineteenth century.

Both of these procedures are much better achieved by clear statements than by the introduction into keys of obscure references to sets of standard terms that few of their users are likely to know anything about.

7 Entropy and liability

Good cryptography usually relies on good entropy (see 2.17). Unfortunately, good entropy is not always easy to come by - it is in particularly short supply on machines that have little interaction with the real world. This is because the usual sources of entropy are things like mouse movements, keystrokes and network events. So, the question is, if your entropy is bad, is it your fault?

The answer to the question depends on what warranties or other contractual terms govern the relationship between the victim and the author or supplier of the software involved. If the problem were explained in the manual, and especially if the user could manually intervene to top up the entropy, there would be no liability. Otherwise there might be, if the procedure used could be shown to be below the standards usually adopted for the purpose; but it would not be an easy case to be confident of winning. Note that entropy or the need for it are not often explained in any end user manual.

Failing to provide good entropy is clearly not wise. Unfortunately, the machines typically most in need of entropy (servers) are the ones least able to find it.

Example: Netscape had a poor random seeding for HTTPS which was successfully attacked [12].

8 CA/RAs and liability

CAs often attempt to disclaim liability for their certificates - can they really do that? Does it make any difference that users of certificates rely on the CA, but those acquiring certificates may deal only with an RA (to whom the CA has delegated part of its function?

The CA's disclaimer works as a matter of the general law, at least if it is effectively incorporated by reference in its certificates. It might be possible in some jurisdictions, such as UK and other EU countries, to challenge the disclaimer as an unfair contract term excluding negligence. But this is no good unless one prove negligence, which would probably not be at all easy in a lot of cases. And it is arguable that a wide disclaimer is in fact wholly reasonable: all a CA can reasonably be expected to say is that it took some specified measures to check that a particular individual was at some moment in the past at least one of the persons who had access to the private key. This can at best provide only modest grounds for concluding that a particular electronic object was signed by any particular person at some later time. If such a conclusion proves unjustified, there may well be no causal connection with any inaccuracy in the certificate, and indeed there may be no such inaccuracy.

If the measures promised by the CA include checking facts with an RA, the CA may well bear no responsibility at law for an inaccuracy which results from an

error by the RA. Even if the RA can be shown to have been at fault, which may be far from easy, it too may be effectively protected by a disclaimer.

The victim is better off if the CA issued a certificate as a "qualified certificate" complying with the EU Signatures Directive, because in a number of cases the CA will have to prove that it wasn't negligent (rather than the burden of proof falling on the victim). This may not help a great deal, however, since the CA may well be able to show that the error occurred despite its procedures being sound and well-documented, and unless the victim can prove that they weren't followed, the CA may still escape.

Sam Goldwyn is reputed to have said that an oral promise wasn't worth the paper it was written on. CA certificates fall into a similar class.

9 Insecure signatures

It is very noticable that the legal and technical concepts of "signature" are wildly different. For example, no techy would consider a scan of your handwriting to be a signature in any meaningful technical sense but the law would clearly recognise it as such.

To understand this disparity, it's important to note that making contracts, the basic element of all commerce, does not in practice depend on secure authentication. Many contracts, indeed the overwhelming majority, do not require writing for their validity, let alone a signature. They can be made by word of mouth, either face to face or by telephone, without a witness or other evidence.

Of course this creates a risk that disputes will be hard to resolve; but mostly it works well, either because there are no disputes, or because the surrounding circumstances leave enough evidence to resolve them, or because the court believes one person's word rather than that of another.

Compared with a contract made on the telephone, a contract made by an exchange of insecurely signed emails already has the great advantage that there is a record of the words used. Compared with such an exchange, the use of electronically signed emails is even better, because it benefits from the assurance of integrity of any signed object (since the signature will not verify if the object is later changed) even if there is little or no technical assurance of authenticity (given the legal problems of relying successfully on certificates).

So, in practice, pressing a button saying "I agree" or even typing your name at the bottom of an email *does* form the basis of a contract, despite techies' revulsion for these methods. The advantage of more technical signatures is simply that it is harder to deny them.

10 Conclusion

We think this paper shows the importance of the continuing mutual education of lawyers and techies: many important contributions by members of either tribe are marred by errors in their understanding of the territory of the other.

References

[1] Section 7(2) of the Electronic Communications Act 2000. http://www.legislation.hmso.gov.uk/acts/acts2000/20000007.htm

[2] The Value Added Tax (Amendment) Regulations 2000 (SI 2000 No. 258). http://www.hmso.gov.uk/si/si2000/20000258.htm

[3] W.Diffie and M.E.Hellman, "New directions in cryptography," IEEE Trans. Inform. Theory, IT-22, 6, 1976, pp.644-654.

[4] Rivest, R. L., Shamir, A., Adleman, L. A., "A method for obtaining digital signatures and public-key cryptosystems" Communications of the ACM, Vol.21, Nr.2, 1978, S.120-126.

[5] Microsoft Security Bulletin MS01-017. http://www.microsoft.com/-technet/treeview/default.asp?url=/technet/security/bulletin/MS01-017.asp

[6] Law Commision, "Electronic Commerce: Formal Requirements in Commercial Transactions", December 2001. http://www.lawcom.gov.uk/library/lcspecial-1/e-commerce.pdf

[7] J. Callas, L. Donnerhacke, H. Finney, R. Thayer, "RFC 2440 - OpenPGP Message Format", November 1998. http://www.ietf.org/rfc/rfc2440.txt

[8] ITU-T Recommendation X.509 (1997 E): Information Technology - Open Systems Interconnection - The Directory: Authentication Framework, June 1997.

[9] A. Frier, P. Karlton, and P. Kocher, "The SSL 3.0 Protocol", Netscape Communications Corp., Nov 18, 1996.

[10] T. Dierks, C. Allen, "RFC 2246 - The TLS Protocol Version 1", January 1999. http://www.ietf.org/rfc/rfc2246.txt

[11] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, T. Berners-Lee, "RFC 2068 - Hypertext Transfer Protocol – HTTP/1.1", January 1997. http://www.ietf.org/rfc/rfc2068.txt

[12] Ian Goldberg and David Wagner, "Randomness and the Netscape Browser", Dr. Dobb's Journal, January 1996, pp. 66–70. http://www.cs.berkeley.edu/ daw/papers/ddj-netscape.html.