DIVISION OF COMPUTER SCIENCE

Authentication v Certification

Marie Rose Low
Bruce Christianson

Technical Report No 216

January 1995

Authentication v Certification

Marie Rose Low and Bruce Christianson

M.R.Low@herts.ac.uk B. Christianson@herts.ac.uk

Computer Science Division,
University of Hertfordshire, Hatfield
Herts, England, Europe

Abstract

Authentication servers and certification authorities are usually viewed simply as alternative ways of providing the same service. In this paper we show that the differences in the natures of these two approaches are more subtle than is commonly assumed. We argue that, because of these differences, there are many benefits to be gained from employing a protection scheme based on a certification authority rather than an authentication server and that these benefits can, in many applications, outweigh the higher costs in performance associated with a public key certification authority.

1 Introduction

Whilst there has to be a strong mechanism for services to protect themselves from malicious users and users from malicious services, any such mechanism must be flexible and independent from as much of the systems' infrastructure as possible if it is to operate successfully over heterogeneous environments and autonomous resource management domains.

The basis of any protection scheme is the scheme's ability to identify correctly the participants in any transaction. The participants usually achieve this by proving knowledge of a secret. In this paper we argue the case for the less popular method of proving such knowledge, that of using public key encryption, as opposed to the more commonly used method of shared key encryption.

It is generally accepted that public key encryption [DiHe76] is well suited to provide digital signatures because it provides attribution of actions. Yet the use of public key encryption has not become widespread. One of the factors that hindered its use is that the only well known algorithm, RSA [RiShAd78], is slow when implemented in software and also has space overheads in the length of the signatures. In addition, protection mechanisms that do use public key encryption have usually been based on the same set of principles and techniques that are used with shared key encryption. This has been a major stumbling block in advancing the use of public keys. Public key encryption is not as efficient as shared key encryption, and when it is used 'instead of' shared keys, there is very little benefit gained.

This is true, for example, when encryption is used to provide confidentiality and data is encrypted before transmission or storage. To ensure integrity data has to be encrypted with the originator's private key and to maintain confidentiality data has to be encrypted with the reader's public key. Two costly encryptions of the data to provide confidentiality and integrity whereas if the two end parties share a secret key then only one, more efficient, encryption is required.

However, the picture changes when considering authentication, authorisation and delegation. We achieve both authentication and authorisation in the real world using our manual signature. To have a digital system capable of replacing the current written signature, it must have the same properties as a written signature i.e. it must be easy for anyone to recognise the signature but impossible for anyone, other than the legitimate user, to reproduce it [DiHe76]. These properties are precisely what public key encryption offers that shared key encryption does not.

2 The Practical Differences

It is important first of all to identify the differences in the functionality of the two schemes. The following sections outline the functions of each scheme and the results of these operations.

2.1 The Authentication Server

Authentication of principals is usually implemented using a shared key encryption algorithm e.g. Kerberos [StNeSc88]. There is typically an authentication server for each management domain. The authentication server shares a secret key with each of the principals in that domain. The authentication server is relied upon to authenticate these principals and their requests. This is done by checking that a principal has knowledge of the shared key and requires at least one on-line transaction with the authentication server for each set of transactions between two principals. The authentication server must be trusted by its clients not to disclose or misuse its knowledge of each client's shared key.

This is even more significant when operating across different management domains, where usually there is a hierarchy of authentication servers across the domains. Then, a client has to trust his authentication server and all those in the hierarchy on a path to the remote party. Thus, in a distributed environment all authentication servers participating in a transaction, whether local or remote, must also be trusted by all clients involved.

2.2 The Certification Authority

When using public key encryption, it is essential that each public key is verifiably bound to the correct owner. A principal achieves this by registering his public key with a recognised authority, called a Certification Authority (CA).

Each principal, including the CA, has a public/private key pair. There is a CA available off-line to each principal in a domain, whose sole purpose is to bind a principal's public key to his name. The CA provides this binding by producing a public key certificate (PKC) for each principal's public key. A PKC consists of the principal's name and his public key, together with an encrypted digest that is dependent on the name, the key and the CA's private key. The clients of a CA must satisfy themselves that the procedures carried out at the CA and the protection of the CA's private key conform with the clients' security requirements. The method for presenting the principal's public key to the CA will depend on each CA's certification procedures. The corresponding private key is never

shown to the CA.

The function of the CA is to generate a PKC for a principal; this is performed off-line and only once[1] per principal. Verification of a PKC may be done locally by any principal, without access to the CA, so that principals become their own authentication servers. All that is needed is a local genuine copy of the public key of the CAs involved and a local trusted version of the public key algorithm.

2.3 Basic Difference

Traditionally, shared key encryption has been coerced into partnership with an authentication server to perform the authentication and authorisation functions necessary to provide the 'equivalent' of a digital signature. When public key encryption has been employed, it has been considered as a substitute for shared key encryption and managed in the same way, as though it needed the equivalent of a shared key authentication server. This is not actually the case; the certification authority required for public key encryption is a different animal to an authentication server. The shared key authentication server verifies all authentication data; the authentication server is the authority that guarantees the validity of authentication data and the server has to be available on-line to and trusted by all who need to use it. The public key certification authority provides an electronic identity for any principal that registers with it, but plays no further part in any verification of data. This verification is done locally by those interested in the validity of the data. This difference in the functionality of the two authorities is clearly illustrated in the paper by Needham and Schroeder [NeSc78] which describes a set of protocols to perform authentication functions using both shared and public key encryption. However, the implications of these different characteristics are not further explored in that paper which concludes that there may, in practice, not be much difference between an implementation of either method. We believe that it is this conclusion which has often led to this paper being cited as proof that there is no particular benefit in the use of public key encryption.

3 Arguments for Public Keys

"Public key techniques offer significant added value by limiting the set of functions which authentication support components must be trusted to perform correctly, and by enabling authentication credentials to be validated throughout a distributed system without recourse to on-line servers capable of impersonating their users" [Linn90].

When comparing schemes, the intended application must be taken into account and the main design issues in constructing a scheme for a distributed system identified [GoBeDa93]. The following issues have to be considered; how many times each authority has to be invoked during an authentication or authorisation process; what the consequences are for the users when the central authority is violated or becomes unavailable; what the risks are to users when trusting a central authority; how two users who trust each other, though not necessarily each other's operating systems, can exploit that trust to enable them to work together.

3.1 Attribution of Actions

The most widely recognised benefit of public key encryption is the fact that any text

---

[1] A principal can of course generate a new public key pair whenever he wants, but within any given set of transactions a PKC need only be generated once for each principal.

signed by a principal with his private key is always attributable to that principal, since no-one else has knowledge of that private key. This is not possible with shared key encryption as there are always at least two parties which share the secret key. By using public key encryption together with a protection scheme such as Self-Authenticating Proxies [LoCh94a] [LoCh94b], a service provider can show the authority under which it performed all transactions by keeping an audit trail of these actions and the signed tokens which authorised them. This authority cannot be forged, and the PKCs can be included in the audit trail and used to verify the signatures. The user of the service can subsequently check that all operations performed have been properly authorised and can hold the service accountable for any violation of trust, and likewise the service can defend itself against false accusations of misbehaviour.

3.2 Local Verification

With public-key encryption, a signed authority for a transaction can be generated locally by a principal and, more importantly, can be verified locally by the service provider using the other principal's PKC, providing that the verifier has a copy of the CA's public key stored locally. The principal requesting the transaction can provide his own PKC along with the signed request, together with copies of any authority tokens invoked, such as capabilities or proxies, and the PKCs required to verify them. As well as being passed over open channels, PKCs can be stored and retreived from an untrusted cache. Neither party requires on-line access to the CA. In the case of a shared-key authentication server, both parties require on-line access. Thus the use of a CA can significantly reduce the number of messages exchanged. This is particularly significant in cross-domain transactions. Public key schemes are particularly suited to distributed service providers where any one of the servers can verify another server's authorisation and PKC. Servers providing the same service on different systems have no trouble honouring requests based on the authority of another server.

It is often argued that an on-line check is still required at the time of verification to ensure that a PKC is still valid. This really depends on the requirements of the application and the revocation methods used. On-line checks can be avoided with a timely endorsement from the CA or from a trusted third party prepared to accept liability[2]. Essentially, the level of assurance of a cache is set by the manager of the cache who can set his own policy according to his needs. Lampson et al in [LABW91] follow a similar principle to enable fast revocation of CA credentials. They propose an on-line local agent which stores principal credentials that have been provided by a CA. The certificates are stored with a shorter life-span than those at the CA. A credential is only valid if it is both authorised by the CA and endorsed by the on-line agent. Since this agent is on-line, the credentials can time-out quickly and be refreshed often. This particular mechanism can easily be generalised.

3.3 A Compromised Server

Existing schemes that are dependent on an authentication server require that the authentication server is trusted by all the principals it serves, as it holds each principal's secret key. Should such a server become malicious or be compromised then all principals who share their secret keys with it are also compromised because the authentication server can masquerade as its clients. This is an important issue even in the case of a trustworthy server, which then has no way to defend its reputation against a malicious charge of

---

[2]This is an extension of Neuman's certified cheque [Neu93].

malfeasance by a dishonest client who repudiates his own (fraudulent) act and falsely attributes it to the server.

If a CA is violated, then the worst it can do is to issue a bogus PKC binding a spurious public key to a genuine principal, or a false PKC for a spurious (non-existent) principal. However, a principal's genuine private key is not compromised when a CA is violated, and a dishonest CA cannot generate a correct transaction on behalf of a principal because the CA never knows any other principal's private key. In particular, a CA responsible for an unauthorised change to a principal's certified public key cannot produce any authority for the change that would survive an audit.

If tokens granting authority to a principal include the signature of that principal's PKC in the text signed by the generator of the token, then a bogus PKC issued for that principal remains useless because it is not bound into any authorising tokens. Acquiring rights for the bogus PKC will typically require the impostor to make cross-domain calls which will leave an audit trail and increase the chance of detection. Also, since a CA must issue PKCs signed with the CA's own private key, it is easy to determine which CA has behaved maliciously and which attempted transactions are bogus. Principals do not need to change their private keys (since these have not been revealed), and no genuine transactions are put in doubt or need to be re-applied. In particular, a principal cannot opportunistically repudiate a genuine transaction following the violation of a CA. This level of protection is not possible with shared key authentication.

A false identity does not do much harm either. A false PKC for a spurious user does not by itself grant the impostor any rights. If a service provider's policy allows it to grant access to any principal holding a valid PKC with no further checks then it does not much matter if the principal is using a false identity as he cannot get any more rights this way than he could get legitimately. A service provider (or any other principal) can endorse PKCs with his own signature for members which he admits to his own 'club' after going through whatever procedures he considers stringent enough. Thereafter he need only rely upon the secrecy of his own private key.

3.4 A Hierarchy of Servers

Authentication schemes such as Kerberos provide user credentials that can be verified by a service provider. Such credentials must therefore be generated by an authentication server with which the service provider shares a secret key. If user credentials are needed for a remote service provider then there typically has to be a cross-domain authenticating transaction between the local and remote authentication servers. This requires a trusted on-line path between the two domains, through which session keys can be exchanged pair-wise along a connecting chain of authentication servers [GoBeDa93]. All the authentication servers in the path have to be available at the time the credentials are generated and all must be trusted not to misuse their ability to eavesdrop or their knowledge of secret keys. Similarly, in the delegation schemes proposed by Sollins [Sollins88], Neuman [Neu93] and Bull [BuGoSo93], there is a need for a trusted path, via a hierarchy, between authentication servers in different domains both to authenticate principals and to verify any cryptographic signatures generated for delegation tokens. A breach of integrity in any authentication server in the hierarchy compromises the entire authentication operation, potentially retrospectively, and an unavailable authentication server causes denial of service.

With a public key encryption scheme following the principles proposed in [LoCh94b], each principal can determine for himself how much he trusts a remote CA. There is, in fact, no need for any CA to have knowledge of or to trust any other CA that is involved in

a transaction.

It is only the principals themselves who need to have confidence in a CA's public key, and they may obtain this confidence in any way which their security policy permits. A principal will usually obtain knowledge of and confidence in a remote CA's public key by getting this key from one or more trusted third parties such as a trusted colleague who knows the key, from a local agent whose business it is to obtain these keys and which is then liable for their correctness, or by a direct physical approach[3]. A highly secure procedure is feasible, since it need be done only once per remote CA. It may be that a particular principal regards the CAs as being arranged in a hierarchy, with each CA acting as a trusted third party for those lower down, but this is a matter of choice for that principal, not a global requirement. Once a remote CA's public key has been obtained, a PKC issued by that CA can be verified locally without the need for secure on-line communications between domains. A principal may even generate his own personal certificate for the remote CA's key so that it may be used with assurance in his future transactions.

There is also no problem with denial of service if a CA becomes unavailable, because there is no requirement for on-line access to a CA even within a single system. CAs can therefore be made more secure by being physically difficult to access, geographically isolated, or accessible only off-line via an extensively monitored channel which may introduce long delays.

There are other advantages in an open environment to avoiding a fixed universal hierarchy of the type required with authentication servers. If a remote user can only be authenticated by his own domain's authentication server then not only must a trustworthy path to that server exist, but the local party must trust the integrity of the remote authentication server, its security policies and its ability to correctly authenticate its clients even though the local party may have no knowledge of the remote authentication service and no means of establishing its correct behaviour. However, each CA can operate its own security policy, and the corresponding security domains do not need to map neatly onto management domains, as is required in the case of a hierarchy. As a CA does not actually authenticate a principal, multiple CAs may be used to provide a principal's PKC. It is the principals who ultimately decide which public keys they will accept. The amount of authority granted by a local principal to clients of a remote CA may therefore be set according to the local principal's level of trust in the remote CA's procedures to generate only genuine PKCs.

3.5 End-to-end verification

Authentication offered by existing schemes is not end-to-end (e.g. client to service provider), but goes from client to service provider to authentication server and back to the client, with a leap to different authentication servers when there is a remote access. A client can never check for himself whether he is communicating with a genuine service or whether the request he issued is the one that was received by the service. This dependence on intermediate entities has problems similar to those associated with a layered approach to communications between two parties, discussed by Saltzer et al [SaReCl84], where upper layers are dependent on and must trust lower layers to perform each operation correctly. An end-to-end check between client and server would automatically authenticate the principals and the authorising tokens to each other. This is possible with a public key encryption scheme because principals are authenticated by their signatures which are

---

[3] Possibly over dinner.

generated and verified locally by the end users. One of the major advantages of end-to-end measures is that individual principals can elect whether to employ them without affecting other systems or users. They are more naturally suited to users' perceptions of their own security requirements because users have to rely only upon their own security components and not on the system infrastructure or autonomous remote domains. Also, the cost of employing end-to-end measures can be directly apportioned to the principals involved [VoKe83]. End-to-end verification enables a coherent means of providing authentication, authorisation and resource management across system and domain boundaries.

4 Arguments Against Public Keys

The element of any key-based scheme that is most open to attack is the secrecy and integrity of each principal's keys. Illicit knowledge of a secret key can bring with it every right owned by the violated party, while substitution of a bogus key can likewise have disastrous effects. It is important to determine and contain the effect that the loss of secrecy or integrity of one of these keys can have on the participants. The computational requirements and performance costs of the public key algorithm are also factors which inhibit the use of public key encryption.

4.1 Cryptanalytic Security of Keys

It has often been argued that the use of public key signature schemes introduces a weakness not present in shared key signature schemes, in that the public nature and large number of PKCs signed by a CA (or of transaction requests signed by a single principal) may provide a known plaintext attack on the private key. (Of course, nobody has admitted to knowing how to effect such an attack in practice on any public key system in actual use such as RSA.)

The security of any particular cryptosystem against cryptanalytic attack is ultimately a matter of faith, but it is perhaps worth mentioning that authentication based upon a shared key scheme such as DES is itself liable to a verifiable plaintext attack as a result of the same considerations. For example, the first argument put forward by Saltzer in [Saltz78], often cited as exposing a weakness of public key signature systems, actually applies to any integrity-preserving authentication scheme, including those based upon shared keys.

A slightly different form of objection concerns containing the consequences of a successful cryptographic attack. In the case of a shared key authentication server, each successful attack allows the attacker to impersonate only one principal, since different principals share different keys with the server. In contrast, in the case of a CA, a single successful attack on the CA's private key would allow any client to be impersonated. We have already largely dealt with this argument in section 3.3, pointing out that the effects of CA violation can be contained, identified and undone by an audit mechanism within a short period of time. However, we make two additional points here. First, in the case of a shared key authentication server, a successful attack on a key shared with another authentication server (such as a parent in the hierarchy) would allow any remote client to be impersonated, without any safeguards. Second, a CA need not (although it typically does) certify all its clients using a single key. However, a CA which uses several private keys would usually be more secure if knowledge of each private key were restricted to a single physical platform. Whether the result is regarded as a single CA or a family of CAs is then a matter of taste, but the key management problem is no worse than for the shared key case.

4.2 Physical Management of Keys

Another group of objections takes the form that the proposed benefits of the public key scheme are illusory. For example the second part of [Saltz78] argues that since human users of any authentication scheme will inevitably make mistakes that could potentially reveal their keys, there must be means for a user to repudiate use of their key, and therefore a user could misuse this method to fraudulently repudiate a genuine transaction, regardless of the underlying cryptographic system. This objection applies to both shared and public key authentication schemes. In the case of a public key scheme, however, the damage is always restricted to the areas where the violated principal can reach and does not extend to any other party i.e. violation of one principal's private key does not allow misrepresentation of any other individual's actions, and the principal himself is the only party who could be responsible for the security breach, since he alone has knowledge of the private key.

This objection can therefore be countered by applying everyday principles of responsibility, so that a user may not invalidate his signature without previous notification and cannot legally repudiate any message signed before he declared his key compromised. In this way a user is motivated both to keep his key securely and to limit his "spending power" to the least privilege he requires, because he is the only one who suffers if his key is compromised. The user thus has an incentive to keep his private keys securely which is not present in the case of shared keys.

Kline and Popek [KlPo79] come to the conclusion that a similar amount of secure mechanism must be trusted for both forms of encryption and propose that the issue of greatest importance is the strength of the algorithm. Whilst there is need for a central authority in both cases, they make no attempt to evaluate the differences in the functionality of each central authority, in the roles that they play and in the trust placed on them in the process of authentication and authorisation.

There are other forms of the argument that public keys provide only illusory benefits, for example that the problems of secure key generation, distribution and management are just as bad for public key Certification Authorities as for shared key authenticaion servers. Certainly extravagant claims have been made for public key systems in the past, to the effect that the use of public keys is a universal panacea that will solve all authentication problems at a stroke, but the fact that these claims cannot possibly be sustained should not blind us to the genuine merits of the public key scheme. The problems of key manipulation for CAs are non-trivial, but they are of a different character to their shared key counterparts, and their solution requires different issues to be addressed.

4.3 Performance

The performance cost of RSA over the shared key algorithm, DES, is high. Computations are slower and the amount of data generated is substantially larger, particularly in any scheme which uses signature blocks as embedded references to other tokens as suggested in section 3.3.

Some applications may not be able to bear these costs and so the public key scheme is not practical in such cases, but in the general world of CSCW, for example, the performance overheads would most likely go unnoticed. An increasing number of applications require a tamper-proof audit trail involving long-term storage of substantial amounts of data. The costs of such storage is likely to continue to reduce. If current trends continue with improvements in network bandwidth and processor power, the effect of the overheads of public key encryption will be reduced. The principles discussed are not dependent on any

particular algorithm, although they are based on the properties displayed by the RSA algorithm. The way is open for a more efficient public key algorithm to be used.

Performance costs may also be reduced by caching tokens such as PKCs in locally trusted storage to avoid the cost of re-verification, by signing digests of transaction sequences up to some pre-agreed "credit-limit" rather than signing individual transactions, and by pre-calculating authority tokens. Pre-planning can also avoid on-line searches for CA public keys. The client can ensure that the CA public key is already in place before making the request. A service, or user, need not obtain a CA's public key directly if the PKC in question is endorsed in some way by a trusted party known to the service.

Although implementation costs have to be considered, it is important that users are provided with tools appropriate to the tasks they wish to carry out, and it is clear that building a protection mechanism with public key encryption provides users and services with a very powerful and flexible tool. Public key encryption is more suitable for authentication and delegation and is not over-expensive when used wisely [ABKL90].

5 Conclusions

"Sound cryptographic and authentication strategies developed in isolation may be defective when embedded in a complete system involving people and legal considerations" [Saltz78].

We have argued that the use of a public key Certification Authority rather than a shared key authentication server confers a number of benefits, not all of them obvious. These benefits arise from the localisation of trust made possible ultimately by the fact that private keys need not be shared with anybody. The benefits include the ability to generate and verify authentication credentials locally, and the fact that servers cannot masquerade as their clients. Since a compromised CA cannot expose any principal's secret key, the effects of such a violation of trust can be contained, identified and undone in a way not possible with shared key encryption. A scheme based upon CAs can provide an unforgable audit trail and does not require global trust in system infrastructure and remote system services nor a fixed hierarchy of trust requiring contiguous management and security domains. There is no need for trusted on-line communications to remote authentication servers, and consequently lack of availability of a CA does not lead to denial of service. We argue that for many applications these benefits outweigh the loss in performance that may be suffered when compared with shared key schemes.

We have shown that an authentication scheme based on a public key certification authority is not just an expensive alternative to one based on a shared key authentication server but that there are essential differences between them. These differences lie not simply in the different algorithm used but arise from the fact that the first scheme provides an electronic identity, ownership of which can be proved by the principal to whom it was issued, whereas the second requires a trusted authority at every point that a principal's identity is to be verified.

We have also argued that providing principals with the equivalent of an electronic identity, a PKC, which they can present as and when required allows users and services operating in an open distributed environment the freedom to set their own levels of trust, their own protection policies, and to use computer systems by following practices much more similar to those employed in everyday life.

References

[ABKL90] Abadi M., Burrows M., Kaufman C., Lampson. Authentication and Delegation with Smart-cards. *Science of Computer Programming*, 21(2):93-113, October 1993.

[BuGoSo93] Bull J., Gong Li., Sollins K. Towards Security in an Open System Federation. In: *European Symposium for Research in Computer Security ESORICS '92*. Toulouse, France, 23 November 1992. Springer-Verlag Lecture Notes in Computer Science.

[DiHe76] Diffie W., Hellman M.E. New Directions in Cryptography. *IEEE Transactions on Information Theory*, 22(6):644-654, November 1976.

[GoBeDa93] Gollman D., Beth T., Damm F. Authentication services in distributed systems. *Computers & Security*, 12(8):753-764, 1993.

[KlPo79] Kline C.S., Popek G.J. Public Key vs. conventional key encryption. In: *AFIPS Conference Proceedings 48*, pages 831-837, AFIPS Press, Arlington, Va., 1979.

[LABW91] Lampson B.W., Abadi M., Burrows M., Wobbler E. Authentication in Distributed Systems:Theory and Practice. *Communications of the ACM, Operating Systems Review*, 25(5):165-182, October 1991.

[Linn90] Linn J. Practical Authentication for Distributed Computing. In: *Proceedings of the IEEE Symposium on Security and Privacy*, pages 31-40, Oakland, CA, USA. 7-9 May 1990. IEEE Computer Society Press.

[LoCh94a] Low M.R., Christianson B. A Technique for authentication, access control and resource management in open distributed systems. *IEE Electronics Letters*, 30(2):124-125, January 1994.

[LoCh94b] Low M.R., Christianson B. Self Authenticating Proxies. *Computer Journal*, 37(5):422-428, October 1994.

[NeSc78] Needham R.M., Schroeder M.D. Using Encryption for Authentication in Large Networks of Computers. *Communications of the ACM* , 21(12):993-999, December 1978.

[Neu93] Neuman B.C. Proxy-Based Authorisation and Accounting for Distributed Systems. In: *Proceedings of the 13th International Conference on Distributed Computing Systems*, pages 283-291, May 1993.

[RiShAd78] Rivest R.L., Shamir A., Adleman L. A Method for Obtaining Digital Signatures and Public Key Cryptosystems. *Communications of the ACM*, 21(2):120-126, February 1978.

[Saltz78] Saltzer J.H. On Digital Signatures. *ACM Operating Systems Review* , 12(2):12-14, 1978.

[SaReCl84] Saltzer J.H., Reed D.P., Clark D.D. End-to End Arguments in System Design. *ACM Transactions on Computer Systems*, 2(4):277-288, November 1984.

[Sollins88] Sollins K.R. Cascaded Authentication. In: *Proceedings of the IEEE Symposium on Security and Privacy*, pages 156-163, Oakland, CA,

USA. April 1988. IEEE Computer Society Press.

[StNeSc88] Steiner J.G., Neuman C., Schiller J.I. Kerberos: An Authentication Service for Open Network Systems. In: *Proceedings of the USENIX Winter Conference 1988*, pages 191-202, Dallas, Texas, Feb 9-12, 1988.

[VoKe83] Voydock V.L., Kent S.T. Security Mechanisms in High-Level Network Protcols. *Computer Surveys*, 15(2):135-171, June 1983.