BSA: A framework for efficient accounting on wide-area networks

Michael Baentsch, Lothar Baum, Georg Molter, Peter Sturm

{baentsch, lbaum, molter, sturm}@informatik.uni-kl.de
AG Systemsoftware, University of Kaiserslautern, P.O. Box 3049, D-67653 Kaiserslautern, Germany

As global networks are being used by more and more people, they are becoming increasingly interesting for commercial applications. The recent success and change in direction of the World-Wide Web is a clear indication for this. However, this success met a largely unprepared communications infrastructure. The Internet as an originally non-profit network did neither offer the security, nor the globally available accounting infrastructure by itself.

These problems were addressed in the recent past, but in a seemingly ad-hoc manner. Several different accounting schemes sensible for only certain types of commercial transactions have been developed, which either seem to neglect the problems of scalability, or trade security for efficiency. Finally, some proposals aim at achieving near perfect security at the expense of efficiency, thus rendering those systems to be of no practical use.

In contrast, this paper presents a suitably configurable scheme for accounting in a general, widely distributed client/server environment. When developing the protocol presented in this paper, special attention has been paid to make this approach work well in the future setting of high-bandwidth, high-latency internets. The developed protocol has been applied to a large-scale distributed application, a WWW-based software development environment.

1. Introduction

Global computer networks like the Internet are more and more attractive not only to private but especially to commercial users. Many companies and organizations have recognized internetworks to be an efficient means to reach potential customers. Unfortunately, up to now most network commerce consists of mere advertisements presented via Web pages. The actual purchase of goods and services over networks has neither appealed to most customers nor to companies, since there is still a need for appropriate, i.e., efficient and non-intrusive security and accounting mechanisms. Several proposals and prototypes addressing these issues have emerged recently [4, 9, 12, 13]. However, most of them are restricted to special usage environments and show a considerable overhead per transaction. Therefore, they seem to be inappropriate for small payments in the order of a fraction of cents which will form a considerable portion of future commercial network transactions. Still others offer efficient accounting schemes but lack strong anonymity. The scheme presented in this paper introduces a security and accounting protocol combining the benefits of the most interesting existing protocols while avoiding their major obstacles.

Originally, this protocol was designed for our WWW-based, distributed development environment BOOSTER [6]. Basically, this environment provides a framework for global software development, distribution, and maintenance via internetworks. As the WWW provides a readily available global infrastructure of servers and clients alike, we chose it to be the basis for BOOSTER. Standard Web servers are used not only as distributed document storage facilities but also for example as the agents for remote compilation and cross-development. The authentication, encryption, and accounting subsystem developed for this environment has accordingly been named *Booster Security and Accounting (BSA)*.

Because of the special properties of the BSA protocol, we believe it to be also suitable for a variety of other WWW-based applications and client/server systems in widely distributed environments in general. One of the key features is its ability to handle sequences of small payments very efficiently, especially with respect to minimal message traffic. No additional message exchange is needed in order to include a complete electronic payment in a normal application level request. Therefore, BSA is especially suited for the future internets in which communication latency between widely distributed participants has to be considered as an increasingly dominating factor when compared to the high speeds of tomorrow's networks. By its design, BSA is capable of off-line operations, i.e., the usage of electronic currency without continuous involvement of a trusted third party. This is especially important if low transaction overheads, good scalability, and minimized effects of network partitionings or host failures are desired. Additionally, the system can guarantee the anonymity of its customers, since the use of BSA's electronic means of payment does not disclose the user's identity. However, besides encrypted message traffic and automatic server authentication, efficient client authentication is offered as an option, thus enabling service providers to implement access control mechanisms.

The subsequent parts of this paper are structured as follows. Section 2 presents the application environment which originally caused us to develop this new accounting scheme. Concept and implementation of BSA constitutes section 3. Section 4 contains a discussion of the BSA design decisions, existing protocol supplements and a list of further issues we are going to work on. The paper concludes after a comparison of our approach with other related security and accounting proposals in section 5.

2. A global application

We consider the availability of a large distributed application to be very important when trying to make experiences and contributions in the area of infrastructures for general networked applications of the future. This is one reason why we created BOOSTER as a distributed framework for global software development, distribution, and maintenance based on the infrastructure of the World-Wide Web.

Technically, unmodified WWW servers are used to hold source code, declarations, object code, documentation, etc. of some system developed online. Access to the documents is provided via Common Gateway Interface (CGI) programs permitting controlled and synchronized retrieval and storage of the given development artifacts. Dependencies between the documents are described within specifically formatted HTML documents that can be maintained automatically via a newly developed, Java-based direct manipulation graphical user interface. This HTML-based approach permits everybody using a common Web client to browse through some project managed by BOOSTER. In order to actively participate on the system, however, a Java-enabled browser is required, permitting users to join project sessions of possibly many different developers. Retrieved code can be modified with a build-in editor, stored back to its server, compiled on a third host, and debugged on possibly yet another machine. Among other things, dependency management, the update mechanism, and concurrent access control is provided

to appear in: Proceedings of IEEE Globecom'96, London, November 1996

by the distributed set of CGI programs on the servers involved.

From an abstract point of view, the implemented components model and structure the artifacts of a system under development by means of a graph interconnecting development artifacts. The notions of *Nodes* and *Edges* as metaclasses form the foundation upon which further BOOSTER components are built. The *SourceCode* and *Library* nodes for example represent C++ source code and declarations, or object code, respectively. Edges like instances of the *DependsUpon* class model relationships between these documents that can automatically be queried, traversed, and acted upon accordingly. For a more general overview of BOOSTER, the interested reader is referred to [6], while [5] gives a more technical report on this system.

With the BOOSTER environment, we believe to have an instrument to check out new concepts in the areas of medium-level communication protocols, cooperative work, large scale software development, and electronic commerce. When considering applications like BOOSTER, some obvious questions have to be addressed. For example, how can precious electronic goods – source code in particular – be protected against any interference like malign modifications while in transit? How to account for the usage of one's machine providing compile services for some remote software project? How to get paid if somebody uses software components developed and distributed this way?

These questions led to the development of a configurable protocol integrating security and accounting aspects, which appears to be also applicable to more general settings than only BOOSTER.

3. The BSA payment protocol

The BSA payment model relies on an infrastructure of banks mutually trusting one another. In contrast to electronic cash systems like Chaum's ecash [1, 2], BSA uses a credit-debit model with banks administering accounts of payers and payees and executing money transfers between those accounts. In addition to that, trustworthy bases are acting as certifying authorities and as arbitration boards in the case of disputes. Since the information administered and needed by banks and trustworthy bases is very similar, the two parties are here combined in the notion of trustworthy banks for simplicity of discussion. Since BSA is using the RSA [18] public key crypto-concept for securing communications channels and performing electronic signatures, banks will also act as key servers for the public keys of their customers in this scenario.

Payments are transacted by the use of special authorization information we call *promissory notes*, which in combination with so-called *cover cards* can preserve the payer's anonymity. Promissory notes are guarantees for a certain amount of money and are signed by the respective trading partners. Cover cards serve the same purpose, but are signed by trustworthy banks which by this means confirm the deposit of a certain amount of money. Payers may write promissory notes up to the previously deposited amount and the use of cover cards enables payees to accept these promissory notes without on-line checking at a bank. Payees can also return a confirmation similar to promissory notes to the customer and may by this means refund the customer or return a bill.

When a customer wants to use BSA for buying goods or to pay for services at a certain vendor, he first needs a cover card as a certificate stating that a bank is willing to pay a certain amount of money to authorized recipients. Although it is possible that banks grant credit to their customers making prepayments obsolete (credit model), we will assume that the customer first has to deposit some amount of money at the bank he has chosen to hold his account (debit model). The electronically signed cover card returned by the bank will then state the deposited amount. This cover card is valid for one vendor exclusively. During the time of the card's validity, the customer may pay the vendor by signing a promissory note stating the sum of the current amount to pay and the accumulated value of all previous payments already made with the corresponding cover. The promissory note is then handed on to the vendor, along with a copy of the cover card. Since the cover card lists the money deposited at the customer's bank, the vendor knows up to which amount to accept promissory notes. An accepted promissory note can afterwards be forwarded to the bank that issued the cover card, which will transfer the due amount to the vendor's account.

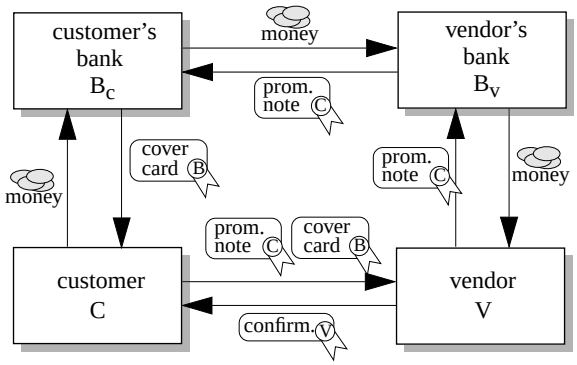

Figure 1 presents *parties and message types involved in a BSA payment transaction. An electronic signature is indicated by a small seal containing the first letter of the signing party.*

At this point, we shall assume to have secure communications channels by employing some kind of encryption, e.g. using the Internet de-facto standard SSL [14] or our combined RSA/IDEA approach as outlined in section 4. Thus we can concentrate on the measures necessary to make the payment system function properly, while preventing fraud by protocol participants. To indicate encryption operations, curly brackets will be used; the text following the closing bracket will denote the key to be applied.

In order to make a payment, customers need a valid cover card. Therefore, the first step consists of depositing real money in exchange for a cover card:

1. C->B_c: pubk_C, VID, money

The message includes the customer's public key (pubk_C), the identity of the vendor (VID) offering the desired service, and in some form the money to be deposited. This step can be executed by physically going to a bank, showing one's public key and depositing real money, or by a short message authorizing the bank to debit the customer's account. In the latter case, additional authentication has to be employed to assure the correctness and freshness of the request. In return, the bank issues a cover card confirming the deposit of the money:

2. C<-B_c: cover_card = {card_id, exp_date, VID, cover, pubk_C}$_{privk\_Bc}$

The cover card consists of a unique card ID, an expiration date, the identity of the vendor for which the card can exclusively be used, the amount of deposited money ("cover"), and the public key of the card's legitimate owner. Including this key makes the card useful only to the owning customer since promissory notes have to be signed with the customer's corresponding private key. It further enables vendors to check the correlation of promissory notes and cover cards off-line.

From now on, the cover card can be used to buy goods or to pay for services at the designated vendor. To make a payment, the customer signs a promissory note and hands it on to the vendor along with a copy of the cover card and the identity of the issuing bank (BID):

3. C->V: BID, cover_card, prom_note_C= {total, sequence_no, card_id}$_{privk\_C}$

The vendor uses the BID to obtain the bank's public key, which is cached for subsequent payments. The cover card is checked for bearing the correct signature and being designated to the vendor. The vendor extracts the customer's public key from the card and checks the signature of the promissory note. The promissory note remains to be the real means of payment: it proves to the vendor (and his bank), that the customer is willing to pay the indicated amount of money. Besides the total sum including all previous payments related to this card, the promissory note also bears a sequence number and the ID of the corresponding cover card. The sequence number is incremented each time customers or vendors issue a new promissory note related to this card, thus indicating the temporal sequence of payment transactions. This kind of timestamp is important in the case of refunds for banks to determine if the customer's or the vendor's promissory note was issued last. Although the sequence number is chosen by normal protocol participants not generally regarded trustworthy, the correct usage of those numbers can easily be checked by the respective counterpart, and incorrect requests will be detected and rejected.

In case the vendor wants to refund the customer, the vendor reduces the amount indicated in the last promissory note received, signs it, and sends this confirmation back to the customer. Since this confirmation will have the same format as promissory notes (except for the different signature), we are using the same term interchangeably for both of them. Thus, promissory notes signed by the customer prove to the vendor (and his bank), that the customer is willing to pay the amount shown in the note. In contrast to that, promissory notes signed by the vendor prove to the customer (and his bank), that the vendor is willing to remit parts of the customer's debts. The same promissory notes can also serve as receipts for the customer, stating the same amount already signed by the customer, or even as bills remaining to be paid (stating an amount higher than the one paid by the customer):

4. C<-V: prom_note_V = $\{\text{total, sequence\_no, card\_id}\}_{\text{privk\_V}}$

With the combination of the messages 3 and 4, the vendor's risk of unpaid order accomplishment and the customer's risk of goods or services held back despite of a payment can be divided and shared by both parties: The vendor can require a first installment included in a request before accomplishing an order, and the customer does not have to pay the complete price in advance so the vendor still has a concern in delivering the goods. Provided that the vendor is willing to wait the time, the remainder can be paid piggy-backed with a subsequent request, so no additional message traffic is needed. In such a case the vendor might want to include a period of time in which he expects the customer to pay. A more sophisticated protocol enhancement for guaranteed delivery of goods will be presented in the following section.

In order to actually receive real money, the vendor applies to his bank handing in promissory notes received from his customers. The vendor may do this after each payment or postpone it to a later time when promissory notes originating from different customers can be bundled. It should be pointed out, that even if the vendor collects a number of promissory notes before handing them in, he only has to send in *one* (the last) promissory note related to each cover card. The vendor's bank then contacts the bank that issued the corresponding cover card and initiates the money transfer to the vendor's account.

When a cover card is about to expire, the customer should return it to the bank to allow a final clearance. The bank will determine how much of the cover deposited was actually spent and credit the remainder to the customer's account. This is also the time the customer can present the promissory notes received from the vendor for proving refunds. In order to prevent repeated usage of cover cards, the vendor will be notified that the card was returned and will thus become invalid. However, the vendor will-have to keep the card's ID until the regular expiration date. Finally, the vendor may hand in the last promissory note related to this card reflecting his point of view.

4. Discussion and protocol supplements

Cover cards were chosen to be valid at one vendor exclusively for the reason of efficient off-line operations. If double spending is to be prevented instead of tracing and afterwards identifying malicious parties, vendor-specific means of payment are the only way to avoid costly on-line validation. In this context, the division of payment data into cover cards and promissory notes serves two purposes. First, the cover card can be used for a series of payments, thus reducing the necessity of frequently obtaining vendor-specific means of payment. The second reason for choosing two separate data items is to preserve the customer's anonymity towards the vendor: neither the promissory note nor the cover card have to carry identity information about the payer for the vendor to accept the payment. The vendor just has to be able to check the customer's signature on the promissory note. This is easily made possible by including the customer's public key into the bank-signed cover card. If it is not possible for a vendor to match a participant's identity to its public key, this method does not disclose the buyers's identity. Additionally, customers may choose to often change their keys. With the separation of banks and trustworthy bases the latter ones are the only authorities which would be able to trace all payments from vendors to customers.

Further advantages of the BSA protocol arise from the ease of fraud prevention and a comparatively small amount of status information the protocol participants have to keep. Each cover card is registered at the issuing bank until it becomes invalid or is returned by its owner. For each card, a log is kept about the sum already paid, so handing in the same promissory note twice does not lead to an additional money transfer. As a result, spending promissory notes multiple times, be it by customers, vendors or by external replay attacks, is uncritical — necessary checks are restricted to the cover cards the promissory notes correspond to. Since cover cards can be issued for high values, the state that banks and vendors have to keep in order to verify payments is very small in comparison to the one incurred by electronic cash systems using the notion of coins.

One of the general problems with off-line electronic payment systems is the missing possibility to assure the atomicity of payment and service accomplishment. This problem is addressed by the BSA scheme, too. With one additional message turn-around, the probability of payments or requested goods being held off can significantly be reduced.

The method is to first deliver requested goods in an encrypted form with only the vendor knowing the corresponding key. The vendor will also add a signed cryptographic checksum over the encrypted data, confirming the delivery of exactly these data. The key to decrypt the data will not be handed over to the customer until the vendor has been paid. The conceptual advantage is, that the confirmation of delivery which was signed by the vendor can be included in the payment (i.e., in the promissory note) as a proof that the overall transaction took place. When the vendor hands in the promissory note, a trusted base can certify that the vendor will have to deliver the corresponding key. When the key is held off, the customer can complain, and appropriate measures, e.g. forcing the vendor to deliver the key, can be taken.

In detail, the extension of the BSA protocol works as follows: The vendor answers a request by delivering the requested data encrypted with a new random symmetric key K, along with the signed result of a cryptographic hash function over the encrypted data (chk_sum).

C<-V: $\{\text{data}\}_{\text{K}}$, $\{\text{chk\_sum}\}_{\text{privk\_V}}$

Having received the requested data, the customer will reply by signing a promissory note and thus paying the vendor. He will do so, because he can not decrypt and use the data. The checksum signed by the vendor will be included in the promissory note as proof of having received exactly *these* data. Afterwards, the vendor can not deny having delivered them, and the customer can not pretend having received different data.

C->V: {total, sequence_no, card_id, $\{chk\_sum\}_{privk\_V}\}_{privk\_C}$

As the vendor is now being paid, he sends the key K to decrypt the data. If the key is held off or a wrong key is delivered, the customer will complain at a trustworthy base, in our case, his bank. When the vendor tries to hand in the corresponding promissory note, the bank sees the proof of payment and data delivery and will demand the key from the vendor. In case the customer complains without reason, the parties will have agreed upon a certain period of time in which the vendors keeps his knowledge of key K. Within this period, the vendor can repeatedly be asked to deliver the key, which should be of no harm to the vendor.

As already mentioned, the BSA scheme also addresses security aspects like secure communications channels, server authentication, and optional client authentication. Since public key algorithms like RSA tend to be expensive in comparison to usual symmetric key algorithms, RSA is only used to exchange a symmetric IDEA [17] session key as commonly done in this situation.

Since the session key can only be extracted by the server itself and since the client will choose a new random session key with each message, giving a correctly encrypted answer authenticates the server and assures the freshness of the answer. By this means, automatic server authentication comes for free.

Besides the wish for anonymous operations, authentication might be desired in some other scenarios, e.g. to implement access control or to allow for customer specific pricing. In such a case, BSA offers clients to include an identification part into their messages. It consists of two items: a certificate signed by a trusted base stating both the client's identity (CID) and the client's public key, and the client's proof of possession of the corresponding private key. This proof is given by the client encrypting some data with his private key. In order to prevent theft and abuse of the identification part, the encrypted data proving the client's identity must include some recipient specific data, e.g. the recipient's identity RID:

ID-part = $\{CID, pubk\_C\}_{privk\_Bc}, \{RID\}_{privk\_C}$

By this means, the identification part is only valid at this specific recipient, and the receiver can not abuse the ID to fake the client's identity himself. The problem of BSA service providers being able to identify the client even without an actual ID once a previous authentication has shown the correspondence of the client's identity and public key is addressed by using different private/public key pairs for authentication, resp. payment or encryption. Additionally, clients may often change their keys. With this scheme, a very efficient client authentication is realized without the need for an additional message turn-around.

On the client side, the computational costs of making a payment are mainly defined by the costs of generating a digital signature on a newly created promissory note. With our current SSLeay-based implementation using RSA private key encryption for digital signatures, this takes from about 10 msec for 128 bit keys up to 100 msec for 512 bit keys on a SparcStation2 running SunOS 4.1.3. Since public/private key pairs are chosen with respect to accelerating public key operations in favor of private key operations, on server side only 5 to 50 msec (depending on the key size) are needed for decryption of the cover card and the promissory note each. About the same values apply to including and checking the identification part in the case of client authentication.

Since cryptographic operations will gain increasing importance in future network access, faster implementations of cryptographic libraries or even special hardware support are likely to be introduced accordingly. In this scenario, BSA takes full advantage of the possibility to piggy-back payment and authentication information directly with any user level request without the need for additional messages. E.g., the payment part of a request could be encrypted in parallel to preparing and sending the request. Another important efficiency advantage becomes obvious in future wide area networks with greatly improved bandwidth but constant latency. With its minimized message traffic, BSA reduces the latency factor to the absolute minimum possible.

An important point for the improvement of this approach will be the explicit distinction between actual money-makers and trusted entities managing the key distribution, e.g. For this purpose, we are currently investigating ways to using a newly developed infrastructure for application-level resource location on the Web. The Web Location and Information Services (WLIS) as introduced in [7] is intended to become an application-level directory service for the WWW and as such provides the needed infrastructure for key management and distribution not discussed within this paper for reasons of brevity. As far as projected work for upcoming protocol refinements is concerned, we will try to further reduce the absolute overhead of BSA. Especially the processing of the RSA encrypted cover cards introduces significant costs. Since cover cards might not have to be kept secret in most circumstances, a simple cryptographic checksum, again RSA-signed by the bank, should also be sufficient as a further configurable option within BSA.

5. Related work

Since the demand for commercial transactions through electronic media became obvious, a plethora of payment schemes, protocols, cryptographic algorithms and even specialized hardware have been proposed [1, 8, 10, 12, 13, 14]. Among the first and most recognized was Chaum's ecash [1], an electronic cash system with separate data items (electronic coins) corresponding to different cash values. With the usage of cryptographic blind signatures [3], unconditional anonymity can be guaranteed to the customer: no one but the customer himself can trace a payment from the payee back to the paying customer. However, payees need to have a bank checking coins for double spending and thus can not accept a payment off-line. Moreover, the databases banks have to keep, logging all coins deposited in order to detect double spending, are continuously growing.

Other protocols follow the credit/debit approach, where authorization information like conventional cheques is used as means of payment. A direct implementation of this idea is presented with NetCheque [13], where payers sign electronic cheques stating the amount to pay and the identity of payer and payee. In this case, one can possibly go without on-line checking since cheating participants can be identified afterwards. However, having no possibility of anonymous transactions might not be very appealing to most customers.

A protocol especially designed for small payments with high frequency is Millicent [12]. It was optimized to create minimal message traffic and to allow efficient off-line checking of payments. The basic idea is to let the vendor himself issue the electronic currency so he can validate a payment without the assistance of another party. In order to pay a vendor, a customer first has to buy a *scrip* from the vendor that can be used only once and exclusively for paying this one vendor. In return to each payment with a scrip, the vendor issues a new scrip with a value reduced by the actual payment. At this point no special care is taken for the case of cheating vendors, deducting a higher amount or not issuing a new scrip at all. This is considered to be sufficient since scrips are only used at low values comparable to the change in one's purse. With the introduction of *brokers* as accounting

intermediaries, anonymous transactions can also be made. No computationally expensive encryption is performed, but all signatures are realized by a combination of shared secrets and cryptographic hash functions. All these measures let Millicent appear especially suitable for micro-payments but also restrict it to the area of low-value transactions, because some security problems are not addressed. Since the vendor always has to return a new scrip, it is impossible to make a prepaid connectionless request as feasible with BSA.

In contrast to that, the NetBill [9, 10] approach puts the emphasis on a high level of security. It is one of the few protocols also trying to couple the payment with the delivery of the requested goods so none can be held off causing losses to the respective other party. In this aspect, BSA's guaranteed delivery of goods as presented in section 4 is actually very similar to the NetBill approach, although the usage of a single NetBill accounting server introduces scalability problems. Finally, NetBill is not flexible enough to reduce the high overhead per transaction when used for small payments.

There are also exist more theoretical advances in this field [15, 16]. The authors of [15] e.g. present an electronic cash system capable of unconditional anonymous off-line payments. They provide an electronic currency that is not only divisible into pieces as small as the payer needs them (as commonly found in most credit/debit systems), but also transferable, so the payee can use a payment received for his own purposes to buy goods at other vendors (as found in most electronic cash systems). Unfortunately, the scheme involves considerably high computational costs and large amounts of data to be transmitted per transaction, so it probably has to be realized in hardware for a sensible applicability.

BSA tries to combine the benefits of most of the protocols presented above. The basic idea is similar to that of Millicent, adding the flexibility and security to handle high-value transactions as well, adding anonymity (although not as strong as that of ecash) and the guaranteed delivery of goods as addressed by NetBill. The computational costs are kept considerably low in comparison to [15], and the additional networking overhead also amounts just to a mere hundred bytes per transaction while keeping the message count to the theoretical minimum.

6. Conclusion

While the focus of developing large scale software systems is more and more concentrating on distributed and networked applications, security issues have gained great importance. It is foreseeable that in the near future network accounting schemes will gain a similar significance, extending the scope of today's electronic commerce to accounting for all kinds of services ranging from providing Web pages to generally using remote computing power, e.g. In this sense, accounting can be seen as an important building block for future distributed applications.

In this paper, we describe an accounting protocol for globally distributed services, which is widely applicable and flexible enough to be useful for various future networked applications. It is both aimed at accounting for small values as well as it allows transactions for precious goods, it enables anonymity where desired as well as authentication where necessary. We deem these features —in addition to the possibility to freely combine them— to be of utmost importance for any generally applicable accounting middleware over basically insecure communications channels. The latter will continue to be pervasive and might in fact even become more widespread due to the increasing introduction of non-stationary, interceptable communication media.

The BSA protocol has been designed and applied in the context of BOOSTER, a globally distributed software development environment using the client/server infrastructure of the World-Wide Web. Special attention has been paid to make the protocol efficient with respect to the number of messages exchanged between participating entities. This is especially important for the Internet of the future, where the relatively large latencies as compared to the increasing bandwidths will be a dominating factor in the design of distributed systems.

7. References

[1] Chaum, D.; Fiat, A.; Naor, N.: *Untraceable electronic cash*, Proceedings of CRYPTO'88, 1988.

[2] Chaum, D.: *Security without identification: Transaction systems to make big brother obsolete*; Communications of the ACM, Vol. 28, No. 10, October 1985.

[3] Chaum, D.: *Blind Signatures for Untraceable Payments*; Advances in Cryptology: Proceedings of CRYPTO'82, Plenum Press, 1983.

[4] DigiCash Corporation: *DigiCash - Numbers that are Money*, DigiCash Publications, 1993. URL: http://digicash.support.nl.

[5] Baentsch, M.; Molter, G.; Sturm, P.: *WebMake: Integrating distributed software development in a structure-enhanced Web*, Computer Networks and ISDN Systems, Vol. 27, No. 6, Elsevier 1995; URL: http://www.igd.fhg.de/www/www95/papers/51/WebMake/WebMake.html.

[6] Baentsch, M.; Molter, G.; Sturm, P.: *Booster: A WWW-based prototype of the Global Software Highway*, Proceedings of the 2nd International Workshop on Services in Distributed and Networked Environments SDNE'95; IEEE Computer Society Press, 1995.

[7] Baentsch, M.; Molter, G.; Sturm, P.: *Introducing Application-level Replication and Naming into today's Web*, Computer Networks and ISDN Systems, Vol. 28, No. 6, Elsevier 1996.

[8] Brands, S: *Electronic Cash on the Internet*, Proceedings of the Internet Society 1995 Symposium on Network and Distributed Systems Security, URL: ftp://ftp.uni-stuttgart.de/pub/doc/security/electronic-cash-on-the-internet.ps.gz.

[9] Cox, B.; Sirbu, M.; Tygar, J.D.: *NetBill Security and Transaction Protocol*, Proceedings of the 1st USENIX Workshop on Electronic Commerce 1995, URL: http://www.ini.cmu.edu/NETBILL.

[10] Cox, B.; Sirbu, M.; Tygar, J.D.: *NetBill: An Internet Commerce System Optimized for Network Delivered Services*, 1995, URL: ftp://www.ini.cmu.edu/NETBILL/publications/CompCon.ps.Z.

[11] Hudson, T.: *SSleay and SSLapps*, URL: http://www.psy.uq.oz.au/~ftp/Crypto/.

[12] Manasse, M. et al.: *The Millicent Protocols for Electronic Commerce*, Proceedings of the 1st USENIX workshop on electronic commerce, 1995, URL: http://www.usenix.org/publications/library/proceedings/ec95/manasse.html.

[13] Medvinsky, G.; Neuman, C.: *Requirements for Network Payment: The NetCheque Perspective*, Proceedings of IEEE Compcon'95, URL: ftp://nii.isi.edu/pub/papers/security/netcheque-requirements-compcon95.ps.

[14] Netscape Inc.: *The SSL Protocol*, 1995, URL: http://home.mcom.com/newsref/std/SSL.html.

[15] Ohta, K.; Okamoto, T.: *Universal Electronic Cash*, Advances in Cryptology, Proceedings of CRYPTO'91.

[16] Ohta, K.; Okamoto, T.: *Disposable Zero-Knowledge Authentication and Their Applications to Untraceable Electronic Cash*, Advances in Cryptology, Proceedings of CRYPTO'89.

[17] Lai, X.; Massey, L.; Murphy, S.: *A Proposal for a New Block Encryption Standard*, Advances in Cryptology, Proceedings of EUCRYPT'90.

[18] Rivest, R.; Shamir, A.; Adleman, L.: *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems*, Communications of the ACM, Vol. 21, Nr. 2, 1978.