RZ 3070 (#93116) 26/10/1998
Computer Science/Mathematics 20 pages

Research Report

Open Issues in Secure Electronic Commerce

Michael Waidner
Zurich Research Laboratory
Säumerstrasse 4
CH-8803 Rüschlikon
Switzerland
wmi@zurich.ibm.com

LIMITED DISTRIBUTION NOTICE

This report has been submitted for publication outside of IBM and will probably be copyrighted if accepted for publication. It has been issued as a Research Report for early dissemination of its contents. In view of the transfer of copyright to the outside publisher, its distribution outside of IBM prior to publication should be limited to peer communications and specific requests. After outside publication, requests should be filled only by reprints or legally obtained copies of the article (e.g., payment of royalties).

IBM **Research Division**
Almaden · Austin · Beijing · Haifa · T.J. Watson · Tokyo · Zurich

Open Issues in Secure Electronic Commerce*

Michael Waidner
IBM Zurich Research Laboratory

October 26, 1998

Abstract

Doing some electronic business on the Internet is already an easy task today. As is cheating and snooping. Several reasons contribute to this insecurity: The Internet does not offer much security per-se. Eavesdropping and acting under false identity is simple. Popular PC operating systems offer little or no security against viri or other malicious software. At the same time, user awareness for security risks is threateningly low.

A few specific security tools are in wide use, and several projects have been aiming at more comprehensive security for electronic commerce. Still there are a lot of important open issues. This article reviews these open issues in a structured way. It is based on experience with the European R&D project *SEMPER*, but it is not a presentation of the project results beyond some useful concrete background for the more general open issues.

1 Introduction

Since the invention of the World Wide Web (WWW) in 1991, Internet-based electronic commerce has been transformed from a mere idea into reality. Consumers browse through catalogues, searching for best offers, order goods, and pay them electronically. Information services can be subscribed online, and many newspapers and scientific journals are even readable via the Internet. Most financial institutions have some sort of online presence, allowing their customers to access and manage their accounts, make financial transactions, trade stocks, and so forth. Electronic mails are exchanged within and between enterprises, and often already replace fax copies. Soon there is arguably no enterprise left that has no Internet presence, if only for advertisement reasons. In early 1998 more than 2 million web servers were connected to the Internet, and more than 30 million host computers [Zeko98]. And even if actual Internet business is still marginal: the expectations are high. For instance, Anderson Consulting predicts Internet business to grow from $10 billion in 1998 to $500 billion in 2002.

Thus, doing some electronic business on the Internet is already an easy task. As is cheating and snooping. Several reasons contribute to this insecurity: The Internet does not offer much security per-se. Eavesdropping and acting under false identity is simple. Stealing data is undetectable in most cases. Popular PC operating systems offer little or no security against viri or other malicious software, which means that users cannot even trust the information displayed on their own screens [PPSW97]. At the same time, user awareness for security risks is threateningly low.

A few specific security tools are in wide use [FoBa97, ECBS97], and several projects have been aiming at more comprehensive security for electronic commerce [Inte97, JECF98, Waid96]. Still there are a lot of important open issues. This article reviews these open issues in a structured way. It is based on experience with the European R&D project *SEMPER* [Waid96, SEMP1398], but it is not a presentation of the project results beyond some useful concrete background for the more general open issues.

*Parts of this report will be published as Chapter 15 of the Final Report of the ACTS Project AC026, *SEMPER*. Our work has been supported by *SEMPER*. However, it represents the view of the author. *SEMPER* is part of the Advanced Communication Technologies and Services (ACTS) research program established by the European Commission, Directorate General XIII. The project was partially funded by the European Commission and the Swiss Department for Education and Science.

Outline: Section 2 gives an overview of the architecture proposed by *SEMPER*, as far as this is necessary to understand the subsequent sections. Section 3 discusses non-technical issues, such as insufficient education of users and the impact of crypto regulations. Section 4 discusses issues that are related to services and protocols, while Section 5 discusses issues related to the implementation. Section 6 summarizes the most important open questions.

2 Overview of *SEMPER*

SEMPER is a joint research and development project aiming at a comprehensive approach to secure electronic commerce. It is partially funded by the European Union and Switzerland.[1] The project started in September 1995 and will terminate end of 1998. This section provides a brief summary of the architecture proposed by *SEMPER*.

2.1 Motivation

Our goal in *SEMPER* has been to design and prototype an open, extensible architecture for the security services needed in electronic commerce. The design is based on a pretty classical notion of a "marketplace:"

- Sellers and buyers have to *find* each other, and have to *negotiate* the *terms and conditions* and *details* of their business. This might result in a *signed contract,* i.e., in a fair exchange of matching offers and orders.
- They have to fulfill their contract, which typically means that sellers *deliver* their goods and buyers make *payments.* Together this is a two-party *fair exchange.*
- Often certain *certificates* and *credentials* will be required in order to check whether a certain transaction should happen or not. For instance, a buyer might require certain *recommendations* by trusted third parties before dealing with an unknown seller, and sellers might require a *proof of creditworthiness* from a prospective buyer.
- Buyers or sellers might be dissatisfied with what has happened so far, e.g., because of errors or fraud. This requires support for *exception handling* and *disputes*, which may involve an *arbiter.*

Buyers and sellers have certain high-level security requirements, like "I don't want to lose money," or "I don't want to see my reputation destroyed," or "I want my privacy to be protected," or "If someone cheated me I want to be able to go to court and win." Of course, for a technical solution this must be reduced to more specific requirements on specific services, and finally results in quite complicated specifications of secure systems (for an example see [PfWa96]).

SEMPER provides a kind of toolbox that allows to implement typical business processes such that security requirements can be satisfied, in a *multi-party secure* way (see Section 4.4.1). The design does not distinguish between sellers and buyers, i.e., it is fully symmetric.

In the following section we will briefly review the service architecture of *SEMPER.* The implementation architecture (i.e., how these services are actually implemented) is not important for the subsequent discussion.

2.2 Service Architecture of *SEMPER*

2.2.1 Service Layers

The *SEMPER* architecture is structured into 5 layers. From the lowest to the highest (Figure 1) these are:

[1] The members of the *SEMPER* consortium are Commerzbank (D), Cryptomathic (DK), DigiCash (NL), EUROCOM EXPERTISE (GR), Europay International (B), FOGRA Forschungsgesellschaft Druck (D), GMD – German National Research Center for Information Technology (D), IBM (CH, F), INTRACOM (GR), KPN Research (NL), Maris (NL), Otto-Versand (D), r3 security engineering (CH), CNET (F), SINTEF (N), Stichting Mathematisch Centrum / CWI (NL), and the Universities of Dortmund, Freiburg, and Saarbrücken (D). The IBM Zurich Research Laboratory provides the technical leadership for the project.

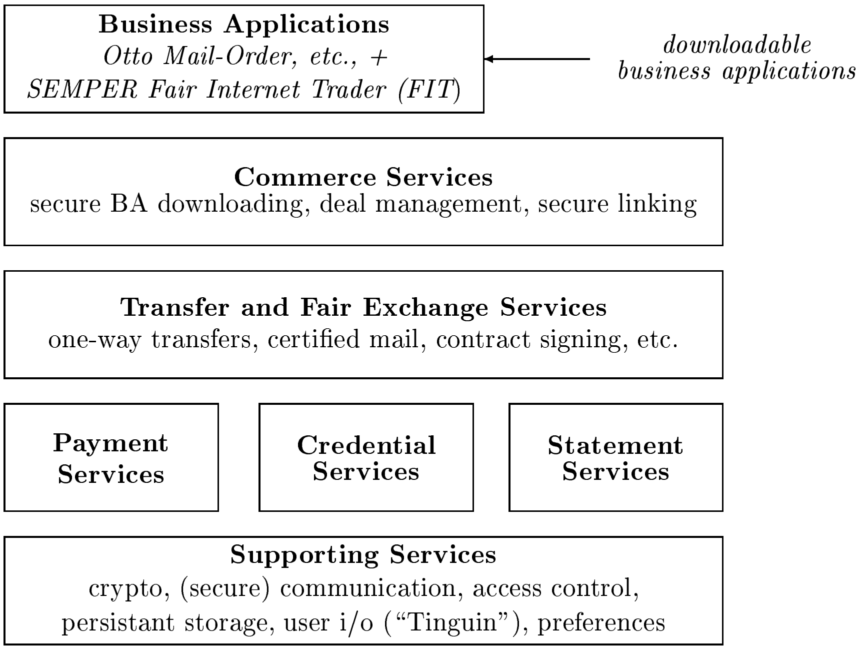

Figure 1: *SEMPER* Architecture

- The *Supporting Services* are the usual cryptographic services, communication, archiving of data (keys, non-repudiation tokens, audit trail), setting preferences, access control, and services supporting a secure user interface with consistent look-and-feel. Furthermore, it provides secure communication services implementing security attributes such as anonymity which must be per connection, not per data item.

- The next layer already contains services that are somewhat specific for commerce applications. It supports making and receiving *electronic payments,* managing and using *electronic credentials and certificates,* and the transfer of any kind of encrypted or authentiated (signed) *electronic statements.*

- The *Transfer and Fair Exchange Layer* contains services for grouping the "items" of the lower layer, i.e., payments, credentials, and statements.

  Such groups can be transferred from one party to another, basically by using the specific services of the lower layer. The user of a transfer service can specify certain security attributes (such as "non-repudiation of origin" or "anonymity") that must be satisfied by the transfer.

  In addition, two items can be *exchanged* in a fair way, i.e., two transfers are combined in an atomic action. Either both transfers succeed, or nothing is transferred at all. The specific fair exchange protocols are located on this layer.

- The *Commerce Layer* combines several transfers and exchanges into a *deal*. A deal is the trace of the actions of a business application, and provides a common context for all actions. Such a common context is particularly important for security requirements that must be enforced across all steps, such as anonymity.

  Preferences specific for a certain type of business, or a certain partner, are defined on this layer. The commerce layer also supports secure downloading of new applications of the highest layer.

- The *Business Applications* implement the specific business processes, e.g., "mail-order retailing," and are the machines that generate the deals managed on the Commerce Layer. This is the most dynamic part of the architecture, and more than in other layers we expect that new, specific business applications will have to be installed on demand.

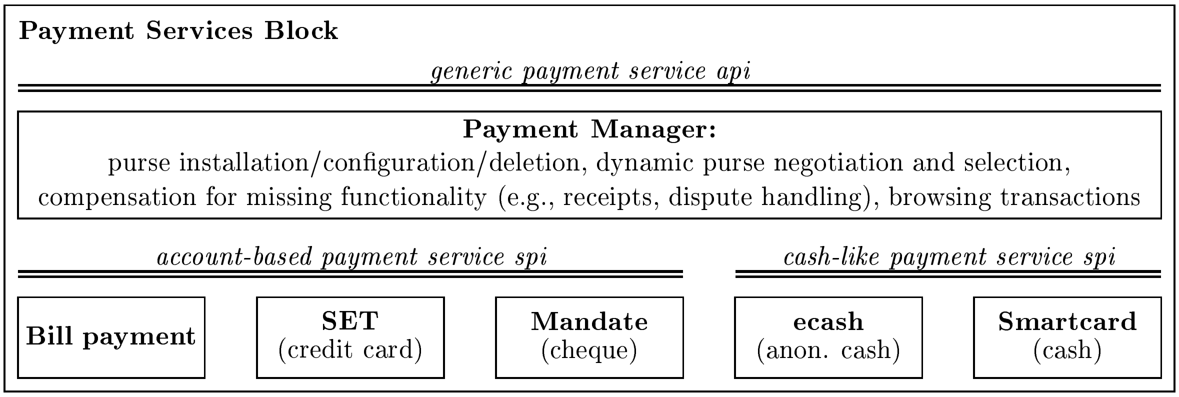

Figure 2: Payment framework

We separated business applications and deals in order get a clearer trust perimeter: Up to and including the Commerce Layer, the system is trusted. Business applications are a priori not trusted. Therefore the Commerce Layer provides a kind of controlled gate for business applications to the lower layer services, and ensures that security critical actions of a business application are always subject to approval by the human user (unless the user declares that he or she trusts a business application completely, e.g., because it is a small script he or she wrote himself or herself, or because it is certified by a consumer organization.)

SEMPER (as well as some third parties for additional trials) developed several business applications. Most of them implement scenarios where a human user, e.g., a buyer, interacts with a server, e.g., of a merchant.

We also implemented a real peer-to-peer scenario, i.e., one where two human users interact: the *Fair Internet Trader* (FIT). It allows two persons to agree on the terms and conditions of a contract, to sign that contract, and to fulfill it. The tool is broadly applicable, in particular for commerce between small businesses which currently falls somewhat between the client-server catalogue scenarios on the one hand and automated EDI solutions on the other hand which both do not offer much flexibility.

2.2.2 Frameworks

We defined interfaces for each of the services mentioned in Section 2.2.1. For most of these services different protocols, and actually different variants of the services themselves, exist. Therefore we decided to design each service block as a *framework*.

As an example, Figure 2 shows the payment framework defined by *SEMPER*. Each such framework contains one *manager* and several *modules*; in the example this is the payment manager, and the different payment systems listed in Figure 2.

Each framework defines two classes of interfaces:

- The *application programming interface* (api) defines the service that is offered by the framework, e.g., the generic payment service, which offers primitives like "make payment" and "receive payment" and administrational services comprising several modules, like showing the available modules.

- The *service programming interfaces* (spi) define the services the manager expects from modules (in the example called "purses"). For the payment framework these are two interfaces, corresponding to the two different payment models supported by *SEMPER*.

All service requests to a framework are first handled by the manager (although modules might offer more than the generic service of the framework; requests for these are then simply passed on). The manager starts a negotiation with its peer manager, which results in a selection of a pair of corresponding modules. The service itself is then provided by this pair. If a selected module does not provide all required functionality, or does not support a certain security requirement, the manager might *compensate,* i.e., might enhance the module service.

The manager is also responsible for all overall administration functions, like module installation and removal and overall transaction browsing.

2.2.3 More Information

More information on the design of *SEMPER*, including the trials and the prototype, are available from `http://www.semper.org`. The final report of *SEMPER* [SEMP1398] provides a more detailed overview, in particular also on the non-technical results of *SEMPER*, such as the Secure Electronic Commerce Agreement. Deliverable D10 [SEMP1098] provides a detailed description of the design.

3 Non-technical Issues

3.1 Security Awareness

Most opinion surveys list "insecurity of financial transactions" and "loss of privacy" among the major impediments to electronic commerce. But in fact most users have only vague ideas about the threats and risks, and a very limited understanding of the technical and legal options for minimizing their risk. As a result all kinds of misperceptions exist.

For instance, the card holder's risk in sending his or her credit card number over the Internet is typically *overestimated.* At least as of this writing payments over the Internet are treated like mail-order/telephone-order transactions, which means that the card holder is not liable at all.[2] All risk is with the merchant.

On the other hand, the risks in sending sensitive data in an electronic mail is typically *underestimated.* Probably most users of email know the mere facts: neither confidentiality nor integrity nor availability are guaranteed. But nevertheless many users do not hesitate to send all kind of very personal and sensitive data to their friends or colleagues, unprotected.

Education and creating awareness has to achieve the following goals:

- Users have to learn and internalize the concept of *multi-party security.* They need to understand their own security requirements, that *they* themselves have to decide whom they can reasonably trust for specific requirements, and that they will typically want to be able to hold these parties *accountable* if they abuse this trust. The users have to learn to distinguish between solutions that support their requirements and trust assumptions, and solutions that violate them.
- Unless better abstractions of security services and properties are available (see Section 5.3) this requires learning the basics of security technology. For instance: What are "passphrases" and how must they be handled? What is a "digital signature," and what is the difference to "entering a PIN?" What does "public-key certification" mean, what are the implied trust and liability assumptions? What does it mean that a transaction is "verifiable?" What is "encryption?" What is a "trusted path from the user to the system?"
- Users have to understand the *limitations* of existing or proposed solutions. They have to learn that a certain level of security requires a certain level of complexity and investment, i.e., that security is not for free.

The last item might be illustrated best by some examples: Assume that a customer, *Curt,* in Germany buys something from a merchant, *Merlin,* in the UK, over the web. Merlin's server supports SSL [FrKK96], and thus Curt is likely to consider the transaction secure. But actually the level of security is probably quite low:

[2] Of course, the card holders are supposed to check their statements of account regularly.

- As soon as the browser tells Curt that it is using SSL he assumes that he is talking with the merchant that is mentioned on the web page, i.e., with Merlin. But to be sure, Curt would have to check whether the *certified* identifier of his partner equals the assumed identity, Merlin, and that the certificate has been issued by an authority trusted by Curt. Most users do neither the first nor the second.
- Even if Curt is talking with Merlin, the achieved level of security is probably very weak: US laws prohibit the export of browsers that support strong encryption. Therefore most US companies produce so-called "export versions" of their products that support only encryption schemes which can be broken without too much effort.[3]
- Finally in a physical transaction Curt and Merlin would probably sign a contract which would, e.g., enable a court to resolve disputes, if necessary. Since SSL is on purpose transparent for the application it cannot provide such evidence.

Similar awareness problems arise even if supposedly secure protocols and implementations for the correct purpose are used: For instance, assume Curt and Merlin are using the SET credit-card payment system [SET97]. Again, the level of security is probably not as high as Curt and Merlin might believe (see also [Webe97]):

- Curt is probably using a popular PC operating system – which usually means that any program he ever installed on his PC can manipulate *any* other program, including his SET software. Maybe Curt installed some software that seemingly does only nice and useful things, but actually also tampers with Curt's SET software and creates bogus payments or modifies the amounts shown to Curt for approval. Who knows? Certainly not the ordinary user Curt. And building such Trojan horses is not too difficult if one knows the software and system to be attacked. Often it is sufficient to know just the user interface of the attacked program.
- Merlin might believe that since Curt digitally signs his payment orders in SET Merlin might be safe in case of a dispute. But since Curt can always *claim* that some malicious software faked his signature (as just explained) there are good chances that courts will not hold Curt liable.

A careful and security-conscious user might now decide to never ever install any dubious software on his PC anymore. But firstly this does not really solve the problem: many standard software has security holes, and without a secure operating system there is not much hope to tolerate those holes. And secondly, there is the risk that this attitude actually backfires: users trained to be careful about installing unbundled software on their systems may decide not to install even security software, relying on the much less secure security software that is usally bundled with their system.

Unfortunately, developers of electronic commerce solutions are often as security unaware and ignorant as their prospective users. For instance, still many developers demand that security must be provided by "lower layers" in a "transparent" way. But, for instance, SSL in an "opaque socket integration" does not make any sense in most cases. Security has to be an integral part of the architecture, design, and implementation.

3.2 Crypto Regulations

Several countries regulate the deployment of strong encryption technology by law.[4] For instance, France controls the domestic use of encryption technology, in order to maintain the capability to eavesdrop on the communication of criminals. The USA prohibits the export of strong encryption products for the mass market, for the same reasons as it controls the export of munition.

Such regulations do not discriminate between "good" and "bad" applications, and limit the security of honest citizens and companies to at least the same extent as they limit the security of terrorists and organized crime [HuPf96]. Therefore several governments, in particular the US administration, are willing

[3] There are some non-US tools available that upgrade those products to the full cryptographic strength, e.g., *Fortify*, `http://www.fortify.net`. But they are not commonly deployed, or used for specific applications only (like home banking).

[4] Cryptography that cannot be used for encryption, such as digital signature schemes and hash functions, is usually not controlled.

to relax their crypto regulations, provided access to the encrypted information would still be possible on demand. The idea is to introduce new “Trusted Third Parties” where secret keys must either be escrowed in advance, or can be recovered afterwards [DeBr96].

All these proposals are still heavily contested, for various technical and political reasons: The Trusted Third Parties would be “single points of failure” for everybody’s security, i.e., new and *extremely* attractive targets for attacks [AABB97]. It is questionable whether *any* regulation of encryption technology can be effective in fighting organized crime: tools for strong encryption are publicly available, and steganographic techniques can perfectly conceal the fact that cryptographic techniques are applied [HuPf96].

Many types of commercial transactions require strong confidentiality, which cannot be satisfied in some countries, or across some borders. For instance, consider two large companies that prepare a merger. Clearly their negotiations require top confidentiality. Even the fact that they are preparing the merger, i.e., that they are communicating intensively, will be extremely sensitive. This requires actually services for anonymous communication. Nevertheless using the appropriate cryptographic tools would be illegal in many countries.

Political regulations are not subject to scientific research. But we clearly see the need for an international agreement on a more liberal and consistent regulation of cryptography. Electronic commerce demands strong confidentiality, which can be implemented only by strong encryption schemes.

3.3 Legal Issues

Surveying the open legal problems in electronic commerce is beyond the scope of this article. Chapter 14 of [SEMP1398] provides a comprehensive discussion.

The two most important security-related problems are the following:

- *Liability:* The financial risk[5] of a user in a specific transaction depends on his or her liability. In principle, if a user bears no liability, there is no risk.

  The main issue here is *fairness:* The liability of a user should correspond to the security of his or her technical equipment. For instance, if it is technically trivial to forge the digital signature of a user (e.g., because the computing base is insecure, see Section 5.1) then this party should not be held liable for his or her signatures, in general.

  Currently many discussions on digital signatures and signature laws ignore this problem.

- *Harmonization:* The national laws that regulate electronic commerce over the Internet (like evidential value of digital signatures, consumer protection, copyright protection) are not harmonized, and are partially contradictory. (One side result is that there is no mutual recognition between national PKIs, even where comparable laws exist.)

  SEMPER’s proposal to deal with this problem is the *SEMPER* Electronic Commerce Agreement (SECA). It is an agreement that can be “joined” by everybody, and which regulates most open or unclear liability questions for those parties who signed it. A midterm solution could be to have several agreements like SECA, for the requirements of specific applications, or for special business relationships. A longterm solution, however, should be a harmonized regulation of all aspects by law in all countries. SECA can be used as an interim solution.

4 Architecture and Protocols

Similar to the overall structure of this article, we follow a top-down approach from higher layers and pervasive issues down to primitives.

4.1 Business Applications

Most of the existing research in security for electronic commerce concentrated on the basic building blocks like electronic payment systems or digital signatures. But what is actually needed are comprehensive, secure *processes,* not just secure *steps,* for several reasons:

[5] Note that there are also risks that cannot be translated in financial terms, like loss of privacy.

- Business processes like auctions or mail-order retailing are composed of a multitude of single steps. Only by considering these steps in the overall context of the process one can decide whether the process is correct and secure.
- Consequently, business processes are the entities that need to be trustworthy for users, not just the single steps. It would be quite trivial to combine secure steps into an insecure process.
- Security requirements like "anonymity" can be satisfied only if *all* parts of a process support them. For instance, it is pointless to use an untraceable payment system together with an ordering system that always identifies the payer. Other requirements like automatically checking the correctness of statements of account require even to consider several related processes simultaneously. Somewhere this connection must be documented.
- Linking steps into processes simplifies the authorization by the user. Authorization for one step in a process often implies authorization for certain subsequent steps. Similarly, a user might wish to define a certain policy that determines the conditions under which certain steps are automatically authorized. This might depend on conditions across the basic steps.

Such linking services are provided by the top layer of the *SEMPER* architecture [SEMP1398]. *SEMPER*'s Commerce Layer only contains the most general linking services, without any semantics that is specific to application domains. Linking steps into processes is left to the user or delegated to the business application designer, and establishing trust in these applications is done via checking certificates prior to downloading the application, according to the trust policies set by the user.[6]

The main alternative would be to enrich the Commerce Layer with "primitives," to design a language that allows to build processes out of these primitives, and to check whether certain primitives are linked correctly (e.g., whether the shipping address in a payment is the same as in the corresponding order). All that would need to be standardized. A wide range of primitives could be added: For instance, one could enhance the generic exchange services with specific semantics, such as "certified mail." Services that add transactional concepts to such processes could be added, reusing similar functionality as already present in the basic transfers and exchanges. Finally domain-specific services and protocols could be added, such as those defined in [OBI98, OTP98].

It is still an open, interesting question how comprehensive this approach can become, i.e., whether one can identify a set of primitives that is *small* enough to be implemented in the Commerce Layer (thereby becoming part of the trusted system) while covering a *large* number of all potential applications. *SEMPER* started exploring this approach with the design of a tool for person-to-person electronic commerce, the Fair Internet Trader (FIT) [SEMP1398]. In any case, if this approach does not work, the approach with checking certificates prior to downloading does not work well either – if almost all merchants need their own business application, it is also not possible to get all these applications carefully evaluated and certified, and thus one cannot trust them enough to use lower-layer primitives (like making payments) without explicit user authorization for all these primitives.

4.2 Pervasive Anonymity

Real marketplaces allow users to act *anonymously.* For instance, there is no need to identify yourself in order to browse a catalogue, or to buy a book in a bookstore (at least not if you pay cash). "Loss of privacy" is often mentioned as a serious concern by Internet users, thus one can assume that most users would like to act anonymously on some electronic marketplaces, too.

The technical approach to anonymity is to let people act under *pseudonyms* that do not identify their real-world names or locations.[7] The use of each pseudonym should be minimized because otherwise information

[6] This is pretty much what all similar projects do at the moment. However, note that in contrast to some proposals, certificates signifying trustworthiness of such an application must be distinct from certificates binding a key to the designer and even from saying that the designer is well-known in some role, in particular as a merchant. Otherwise one is back to the situation where merchants can simply take money out of their clients' purses, and all the other security measures could be omitted.

[7] In some cases, this identification might be required, primarily for dispute handling (like in the German signature law). But in many cases even a dispute can be resolved anonymously [PWP87].

accumulates and reidentification often becomes possible. Several protocols for anonymous communication between holders of such pseudonyms are known – [PfWa87] is still a valid overview of the basic concepts. There are also various proposals for anonymous payment systems, ranging from the commercially available *ecash* by Digicash over research prototypes such as *CAFE* [BBCM194] to an endless number of scientific papers (see [BuePf89, PWP87, AJSW97] for some surveys).

SEMPER did some work on integrating such primitives: We integrated *ecash*, and designed and implemented a protocol for untraceable communication [Frit98, SEMP1098], based on the concept of "Mixes" [Chau81].

Several problems are still open, in particular that of *pervasive anonymity* [SEMP1098]: A system like *SEMPER* has to offer a consistent and enforceable abstraction of anonymity, both to developers and to users. From the user's point of view it must be possible to demand anonymity for a specific process such that the system enforces this anonymity across all parts of this process consistently, or at least warns the user if the process will violate the anonymity requirement, or may link different pseudonyms that were so far not linkable.

There are also open problems with the individual primitives. While untraceable payment systems are already sufficiently secure and efficient, in particular for use in the typical online scenario of electronic commerce, this is not true for the two other main classes of primitives, *anonymous communication* and *unlinkable credentials:*

None of the current protocols for anonymous communication is *both* suitable for the Internet *and* offers a predictably high degree of anonymity. For instance, the systems described in [ReRu97, ReSG98] are relatively efficient, but offer only a low degree of anonymity. The systems described in [PfWa87, JMPP98, Abe98, Jako98] offer a predictable degree of anonymity but cannot be deployed across the Internet. More research is needed.

As already said, the basic approach to anonymity is to use pseudonyms instead of names. Technically one can identify a pseudonym with the capability to digitally sign statements for it, and credentials become simply statements about a certain pseudonym, signed by (with) some other pseudonym [PWP87]. As users start using different, unlinkable pseudonyms, the need arises to transfer credentials issued on one pseudonym of a user to another pseudonym – without revealing to the commerce partners seeing these pseudonyms that they belong to the same person [Cha885]. This is called *unlinkability.* There are efficient constructions for basic unlinkable credentials (starting with [Cha885]), but the construction of semantically rich or personal credentials, and of satisfying complex policies is still an open problem.

4.3 Web Tracking, Personalized Accounts, and Direct Marketing

One of the real differences between conventional and electronic commerce is the cost effectiveness of *personalization* of marketing information and intangible goods [Bako98]. For instance, several search engines determine which advertisement to show by analysing the current request: the search result for "Switzerland" is likely to be accompanied by an ad from a travel agency. Other examples include customized stock quotes, or book recommendations based on prior purchases at an Internet bookstore. Another example is spam mails, which are often caused by previous requests to servers.

Certain types of such directed marketing are obviously advantageous for everybody, in particular those that are fully controlled by the customers. But other types can easily become a privacy nightmare. *SEMPER* did not address this problem at all, and actually only little research is going on in this area [CrLa98, GGMM98, BGGM98, DeRi98].

4.4 Public-Key Infrastructures and Trust Management

In order to employ public-key based security solutions one often needs an infrastructure for secure user registration, public-key certification, and directory services. Usually this is called *public key infrastructure* (PKI) [FoBa97]. There are several open problems with the development and use of PKIs:

- Systems for *global* electronic commerce need presumably extremely large PKIs, serving tens of millions of users. So far nobody has built such a system, and doing just this is already a big challenge. Obvious issues are the operational security of such an infrastructure, and the availability of the service.

- PKIs become new but obviously very attractive targets for attacks. For instance, corrupting the certification authority (CA) used by a bank might enable to rob this bank. So far nobody knows how big this risk really is. (More general details on establishing trust in a PKI are discussed in Section 4.4.1.)
- There are several classes of certificates, whose differences are often not understood (in particular as the formats themselves don't always contain the semantics) and among which one has to choose for the initial infrastructure or for each purpose.
  - The most often used standard for public-key certification, X509, interprets a certificate as a statement saying that a certain public key belongs to a certain, distinguished *name.* The intention is that any such distinguished name refers to a certain real world entity, like a human being, an organization, or a machine.
  - The opponents of this approach argue that the concept of a *name* is not appropriate for electronic commerce (cf. [RiLa96, BlFL196] and the IETF SPKI working group). Instead they propose that a certificate should be a statement signed by some authority saying that the owner of a certain secret key (in the certificate identified by the corresponding public key) has a certain *right.* Such rights could be ownership of a credit-card account, the proxy right for a company, or simply the right to access certain information in a database. Such more general certificates are often called *credentials.*

  Currently industry and banking seem to be very much in favour of the more classical X509 approach. For instance, X509 certificates are used in several applications, e.g., SET and SSL, and supported by the IETF PKIX working group. As long as liability with respect to real persons or their external assets is to be created by electronic commerce, at least some use of these ordinary certificates, and thus the non-electronic registration of keys that the second approach tries to avoid, is actually unavoidable.
- Often even the basic semantics of each class of certificates is not clear. For instance, many people believe that an ordinary certificate *creates* liability for the party who owns the secret key that corresponds to this certificate. But a user has to explicitly *agree* on taking such liability during registration, and if a certification authority claims an increased liability for the user without any foundation in the registration contract then this is likely to be legally meaningless.

  Also the "rights" in the second class of certificates are sometimes taken to mean an expression of trust by the certifier, whereas, e.g., the right to use the money on a certain account is simply a matter of correct handling of ownership.
- Often a single credential will not be sufficient to decide whether a certain action is authorized or not. Determining which credentials are needed and creating trust in some entity by looking at its credentials is called *trust management.* We discuss this in some more detail in Section 4.4.2.
- Several proposals exists how to *revoke* public keys, but none is fully accepted. Maybe the safest approach is to always require online verification of certificates, but obviously this is not always feasible. Alternatives like certificate revocation lists are costly and always introduce a certain delay until they become effective.

  An often overlooked problem with revocation of signature keys is that it requires secure *time stamping.* Without time stamping one could use a revoked key forever, by simply backdating signatures.

  Another nasty and not really well-solved problem is how to proceed if a PKI has a root key and that becomes compromised (see [SET97] for a discussion and a first, pragmatic solution).

4.4.1 Creating Trust

In order to do any kind of commerce parties have to *trust* other parties, to some extent, for some time, with respect to some functions. Basically there are three (non-exclusive) approaches how to deal with this necessity:

- One can *minimize* trust, i.e., design systems such that one party does not need to trust other parties unnecessarily. This is the basic concept of *multi-party security.* One particular flavour of this approach is *accountability*, i.e., to rely on the trustworthiness of some party, but ensure that in case this party cheats such misbehaviour would be detectable and provable to an arbiter. In order to discourage such misbehaviour (including negligence that allows employees or even outsiders to carry out the actual misbehaviour) one should maximize the party's risk in this case (punishment, loss of reputation) and minimize the potential gain.

- One can design systems such that trust is *distributed.* Basically, this means that there is no single point of failure in the trust assumptions. For instance, if a certain signature is valid only if the holder can show n certificates then no single certificate authority can cheat the user by issuing a wrong certificate: at least n CAs would have to co-operate to achieve this fraud. To balance security and availability, one can often design k-out-of-n systems, i.e., there are n specific parties and the system is secure as long as at least k of them are honest. A weaker way to distribute trust is to offer at least a certain choice of parties to be trusted (as in real life everybody can choose among many different banks the one that looks most trustworthy – as far as one can see that).

- One can *establish* trust in a particular other party by means of recommendations, and by clearly identifying the scope of trust (with respect to time, functionality, liability, etc.). This is basically the problem of *trust management,* discussed in Section 4.4.2. Still one needs some original trust. This may be improved upon by public control of the respective parties and opening the design of their computer systems to public scrutiny.

These three concepts are well-known in the theory of security systems, but not much use has been made of them in practice so far.

4.4.2 Trust Management

Credentials or certificates are used to decide whether a certain action (typically designated in a message signed with the certified key) is authorized or not. This ranges from the simple question of deciding whether somebody is believed to be who he claims when sending the message, over classical access control applied to decentralized systems, to the installation of a certain program based on its origin. The set of necessary credentials might be clear to the requestor a priori, but in the general case it is negotiated between requestor and the party that manages the resource. Since the decision might involve credentials signed by different parties with different validity rules (like mandatory online checking) this negotiation might actually involve more parties than just the requestor and the resource manager. The whole problem area is collectively called *trust management* [BlFL196].

Some sophisticated proposals exist already [BlFL196, BlFK98] but still many practical problems are unsolved, or at least it is not clear whether the proposed solutions are sufficient:

- Most of the proposed applications for trust management assume a cross-organizational, role-based model. So far there are not many examples of such models.

- Trust management provides an *application independent* mechanism for access control decisions. The alternative is to let each application implement whatever mechanism it needs, and provide only an interface that supports retrieval and verification of credentials. This is pretty much the classical approach, and it is still heavily contested which alternative is the best [Feig98].

- So far trust management systems did not consider *unlinkable* credentials, which means that they provide only a very limited form of anonymity.

Sometimes trust management is not based on and aiming at binary trust decisions, but *quantitative measures of trust.* This somehow captures the idea that trust is the *probability* that a certain party behaves as expected. Most proposals in this area are somewhat ad hoc and lack a real statistical semantics (see [Maur196] for more details). So far there is no practically relevant *and* mathematically sound proposal for a measure of "trust" (and similarly of "security").

4.5 Electronic Notary

Traditionally, the term "electronic notary" means all technical services that one can reasonably identify with services provided by existing, real-life notaries. Examples are notarization of signatures [AdZu97], certified mail, time stamping [HaSt91, BaHS93, BLLW98, ACPZ98], secure archiving of documents[8], but also some support for dispute resolution. Some notary services are already commercially available[9] but the main issue is that most such services are still not well investigated.

Technically, *fair exchanges* are at the core of many notary services, e.g., certified mail and contract signing. See [SEMP1398] for *SEMPER*'s results in this area.

4.6 Multi-party Protocols

Most problems in electronic commerce involve only two users, besides a number of enabling parties (like banks, directories, certification authorities). But some problems naturally generalize to an arbitrary number of parties, for instance, *auctions* [Turb97] and *voting.*

SEMPER did not work on such n-party protocols although they would naturally fit into the architecture. For most problems there are actually no efficient *and* multi-party secure solutions known yet (see [FrRe95, CFSY96] for state-of-the-art protocols). Most practically deployed solutions are *not* multi-party secure, as they are based on a central, trusted party, like an auctioneer or a tally center.

4.7 Dispute Handling

Systems for electronic commerce have to be prepared for the case of disputes among the parties . This means that firstly, the protocols must generate all necessary evidence, typically in the form of digitally signed documents and receipts. Secondly, the system must be able to *evaluate* this evidence, i.e., there must be protocols and procedures that can be used to prove certain facts about transactions to a third party, e.g., a court. Thirdly, there should be a way to express dispute claims in a way that does not depend on the specific protocol, i.e., a generic dispute language is needed.

This does not mean that an electronic commerce system must enable automatic resolution of disputes. Finally, legal disputes will have to be resolved by human judges. But the system must prevent unnecessary disputes (by enabling the parties to check what evidence they have to prove a certain claim), and it must assist human expert witnesses in their analysis of a certain transaction.

SEMPER did some work on this topic, in particular in the context of electronic payment systems [AsVS98]. But still it is an open issue how to describe and resolve dispute claims in general, for all kinds of commerce services. Finding a solution also requires good formal models and security definitions, and techniques to prove systems secure. A rather practical problem is how to add disputability to existing protocols – note that although many electronic commerce protocols create all necessary evidence, almost none of them supports storage, retrieval, and evaluation of this evidence.

To understand the practical need and problem just imagine the following paper scenario: Consider a company that cuts off all signatures from all letters, contracts, cheques, etc., it receives, and throws them in the waste basket. Nobody would expect such a company to convince a court of anything, in case of a dispute. But this is actually pretty close to what happens in many security systems. For instance, SET does not cut off all signatures, but current implementations do not offer a way to get them back once stored, i.e., they are as useless as they would be in the wastebasket.

4.8 Atomic Secure Transactions

Electronic commerce systems are often implemented using standard approaches to transaction processing, aiming at atomicity of commercial transactions. For instance, a payment shall either complete successfully for payer *and* payee, or shall not happen at all.

[8]The main issues are *availability* of the stored data for a very long time, and *confidentiality* of the stored data. Note that many cryptographic keys have a lifetime of a few years only.

[9]For instance, time-stamping by *Surety Technologies, Inc.*, `http://www.surety.com`, and certified mail by *e-Parcel, LLC,* `http://www.e-parcel.com`.

Actually, [BuePf89] describes how to achieve atomicity in payment systems in a multi-party secure way. They also demonstrate that in some cases pure atomicity is unachievable and dispute handling must come to the rescue.

But unfortunately, the similarity between database transactions and commercial transactions easily leads to simple minded implementations that do *not* provide the required multi-party security. Often security is implemented using a *centralized* trust model, i.e., there is one entity that ensures security of the whole system. For mutual identification and secure communication protocols like SSL are used.

SEMPER did some research in this area by working on one important primitive for such systems, fair exchanges [SEMP1398, AsSW97, Asok98, AsSW98, AsSW198]. In this context one might interpret fair contract signing as a multi-party secure and verifiable version of distributed commit. But this was really just a starting step; more research is needed in this direction.

4.9 Electronic Payment Systems

Electronic payment systems have been the prime target for research in electronic commerce for over two decades.

If one looks into open issues it is appropriate to distinguish between *untraceable*, privacy-protecting and *traceable* payment systems: for the latter basically all open research questions are solved, and the only big open issues are standardization of protocols, scalability of solutions, and integration in applications. Untraceable payment systems exist, too, as is demonstrated by systems like *ecash* and *CAFE.* The main problem with this type of systems is certainly political acceptance.

Some technical questions are still open, but technically the main question is that of designing a system that is provably secure. This presupposes a formal definition of payment systems, which is by itself a research problem.

We already mentioned the issue of dispute handling in payment systems. *SEMPER* made some proposals in this area [AsVS98], but in general this is a largely overlooked area.

The often discussed risk of money laundry exists but is generally overestimated: For non-anonymous systems the existing control and verification procedures can be used, exactly as in the paper-based world. For anonymous systems the main issue is how to prevent it from being more extensively used than cash today. The basic approach is to limit the maximum amounts of payments, or the amounts that can be withdrawn by users, on a per user basis.

4.10 Cryptographic Primitives

Basically all cryptographic primitives needed for electronic commerce exist, in some way. The open questions are rather in the areas of proofs of security, and secure implementations:

- Development of *efficient* and *provably secure* cryptographic primitives. Today most systems are not both efficient and provably secure. Using provably secure systems reduces the overall risk of failure, and allows one to make clear statements about the security of the system.

- *Secure initialization* of cryptographic primitives, which means primarily the secure generation of secret keys. Note that many systems using cryptography were broken because the pseudorandom number generators used were predictable [EaCS94].[10]

- *Variants secure against non-functional attacks.* This means variants that do not leak information on secret keys via hidden channels. Such hidden channels can be created on purpose by an attacker (Trojan horse attacks) or might exist because the time to compute the primitive or the power consumption depends on the secret key in some specific way [Koch96]. Some hidden channels can be exploited only in case of a failure, as the attack depends on the specific way the implementation reacts on a failed computation. This includes fault-based cryptanalysis [AnKu96, BoDL97] as well as the usual problems with non-encrypted backup tapes.

[10] In general it is advisable to combine *several* sources of randomness. In particular the system should involve the user into the generation of one source, e.g., by measuring the time between key strokes.

An alternative is to try to destroy the hidden channels in the implementation of the computing base (see Section 5.1), but this is often more difficult than providing more robust algorithms.

- There are *several* standards for all kinds of cryptographic primitives, including message formats and application programming interfaces. This is no serious problem, but having *one* standard would be preferable.

4.11 Intellectual Property Rights

The protection of intellectual property rights (IPR) on the Internet is still an open problem. *SEMPER* did not work on this topic as it was the subject of several other EU projects, and is somewhat orthogonal to the problems solved by *SEMPER*.

5 Implementation

5.1 Trustworthy Computing Base

The most fundamental problem in security in general is that of finding a trustworthy computing base. In contrast to traditional centralized security systems, this is in particular true for the end users' devices because anything a user can do online a successful attacker against these devices can, too.

Today most electronic commerce systems depend on standard PC operating systems and standard web browsers, maybe using scripting languages like Javascript. So far all that has been notoriously insecure. This is in particular true for operating systems that do not support any separation between different processes, which is true for the most popular ones. In such systems any malicious program can completely corrupt any security system, without giving the user a chance to recognize it easily. Such malicious programs can come as part of, e.g., a game or a tricky, seemingly useful utility tool (in which case one calls it a Trojan horse), or as a virus attached to some other program. Anti-virus tools do not detect Trojan horses unless they have been recognized by the manufacturers of these tools.

With Java, attempts have been made to replace parts of the missing operating system functions in a programming language. Combined with a browser allowing applets only in this language one certainly gets a much smaller risk than without such an approach. Still, on the one hand, not all malicious programs come via the browser. On the other hand, some flaws in Java have also been found already, and to some extent Java simply shares the problem of current operating systems that they are big and without clear semantics.

Even if there would be a sufficiently secure operating system there would be the problem of distinguishing between authentic systems and rogue systems. The standard answer to this problem is secure mobile user devices with certain physical, recognizable characteristics. But even then a lot of problems are still open [PPSW97].

For some applications one assumes that certain pieces of hardware would resist any physical attacks, i.e., that they are tamper-resistant or at least tamper-evident (i.e., one can distinguish attacked from integer devices). A lot of research has been done in this area, but only little of it has been published. This means that the general public has only very limited knowledge about the physical security of devices.

For instance one result of this is the perception of smartcards as tamper-resistant devices. Recent experiments have shown that most smartcards offer only poor security against moderately skilled and funded attackers [AnKu96, Koch96]. Better solutions exist[11]. Moreover, of course, the path to the user must be trustworthy, i.e., a smartcard-like device alone cannot play the role of the secure end-user device anyway.

Tamper resistance of devices of parties that need trust by others is an even bigger problem (but of course such devices can be more expensive), because such devices should be strongly tamper-resistant against insider attacks, too.[12] The only existing standard in this area, FIPS PUB 140, only considers restricted classes of attacks. Stronger public standards are needed.

[11] For instance there are a couple of tamper-resistant PC cards. The main reason why they offer better security is their improved coating, and their ability to erase their volatile memory in case of an attack, thanks to an onboard battery.

[12] In contrast, for applications like signatures and online payment, the end user device need not be secure against that user herself.

5.2 Generic Interfaces

SEMPER's primary objective was to provide an architecture that can cover *all* services required for *secure* electronic commerce, and a mechanism to securely combine those services into business processes. We achieved this goal in one specific way, but there were several design questions for which it is difficult to say which decision would have been the "best" one.

The most important decision was to design *SEMPER* as a set of *frameworks* (see Section 2.2.2).

The essential questions with respect to this approach are:

- Are the blocks in *SEMPER* the most useful fundamental service blocks?
- Is it necessary to give all these blocks a framework design or are there some for which no different protocols will co-exist and thus the plug-in concept is an overkill?

The most obvious alternative to what *SEMPER* did would have been to provide only the most basic blocks (e.g., like those supported by [Inte97, JECF98]), and let the users add new services and specific implementations dynamically.[13]

The main argument for such a more limited approach is the doubts that the similarities between related *complex* services are strong enough to lead to efficient and useful block designs. Our experience with the design of a payment framework and also the fair exchanges makes us optimistic that our approach was in fact the better one, or at least that such standardizations can usefully be carried to much higher layers than in most other approaches. But it is hard to find a yardstick for the quality of such decisions.

Another open question is how to structure a system such that it benefits most from secure mobile user devices (which would be similarly powerful as today's PDAs or mobile phones). Unless one can put the whole system on a secure device one needs to identify the most security-critical pieces and squeeze those into the more secure environment first.

5.3 Visualization of Security

From the point of view of an ordinary user, the concepts and options of most security systems look very complex, and are in fact difficult to understand. Even most computer scientists outside the narrow area of computer security and cryptography have difficulties to understand all the possibilities of trust management, and even more the concepts and options an overall system like the one designed by *SEMPER* offers.[14] Nevertheless one needs to interface with the user, showing him security-relevant information and getting his input with respect to functional and security options, and most importantly, his agreement or disagreement with security-critical steps (such as authorizing a payment, or approving that a certain piece of confidential data is sent to someone).

There are (at least) three problems to be solved, in order to visualize security to the user.

5.3.1 Finding Simple Abstractions

The technicalities of how security works are far too complex for ordinary users, thus a simple abstraction is needed. For instance, the intuitive model of a car most drivers have is very different from the model an engineer would use, but still it suffices to operate and use the car safely. Something like this is needed for security, too.

The basic approach we took in *SEMPER* was to channel all communication between the user and the system through a single window, the *Tinguin* (Trusted INteractive Graphical User INterface), and to tell the user that exactly what is communicated through this channel is trustworthy. Nothing else, in particular nothing that is shown on a browser's window. This allowed us to present a consistent look-and-feel for all components.

Within this symbolically confined system, the Tinguin, the security of actions was characterized by letting the user require certain *generic security attributes*, like "confidentiality" or "authentication."

[13] Note that in *SEMPER* a service provider can offer a richer interface than the one required in a block.

[14] And this is not a specific problem of *SEMPER* – if the system as such provides fewer blocks, understanding the overall security including the external services will be even harder.

The prototype showed that channeling everything through one single window was easy to understand and greatly increased security awareness. But our concept of generic security attributes was still too technical, too far away from the ordinary user's world. On the other hand the concept of deciding only between "trusted" and "untrusted" was too simplistic; better, fine grained abstractions must be developed.

5.3.2 Minimizing User Interactions

Given a certain user-comprehensible model of the system one needs to design the dynamics of the user-system interaction. From a security point of view the most important points are the following:

- Most transactions have a few *points of no return,* i.e., steps that if taken cannot be reversed locally. Each such step needs to be authorized by the user, if taken. Examples are sending an email, or concluding a payment. Such points must be clearly indicated and explained. In *SEMPER* we tried to keep the appearance of these points uniform across different applications, thereby teaching the user to be alerted whenever such a point is reached.
- Obviously the power to authorize to take a point of no return must always be with the user. On the other hand users must not be asked to authorize steps too often, as they might then do it without thinking. Thus there must be a way to pre-authorize certain steps, e.g., by deducing authorization from already authorized steps (like 'if the order is authorized then payment is automatically authorized as well'), or from policies ('for the next 30 minutes all payments up to a total of 10 EURO are automatically authorized'), or from preferences ('always send my certificate to merchants who can show a certificate issued by XYZ'). In general context-based, intuitive authoriziation systems are needed.
- Users might be able to manoeuvre their system into obviously dangerous states, like authorizing a priori all payments. The system should not necessarily prevent this completely, but should alert the user and possibly reset the system automatically to a somewhat safer state after a while. The most popular example is to automatically close a user session that has been inactive for a certain time.

5.3.3 Actually Showing it to the User

Finally one needs to map all that to a real user interface. The only security-related difficulty we considered in this part of the design was support for "small" displays: We already discussed that in an ideal world security systems need to be implemented on trusted systems, which are likely to be physically rather small and compact, like a PDA or mobile phone.

We took this into account by keeping the actual design of the Tinguin small and simple, i.e., it would fit on the screen of standard PDAs. We also considered the idea of a "split Tinguin," which means to split the output of a system into "secure output" and "fyi output." The secure output has to be displayed on a trusted system. If the trusted display is powerful enough then the fyi output is displayed on it as well, but otherwise a less secure output channel is used.

The size and quality of the output channel of the trusted device has further implications; see [PPSW97] for a discussion.

6 Summary

In the course of *SEMPER* we identified several open issues and problems in secure electronic commerce. The most important issues are the following (in the same order as they appeared before):

1. Users are not sufficiently aware of security risks and problems, and limitations of existing solutions . This does not only reduce their own security, it also prevents industry from developing really secure solutions as their market chances would be questionable.
2. In many situations it is not clear who bears what liability, and often the assignment of liability is not done in a fair way, respectively in a way compatible with the technical situation.

3. Electronic commerce is about business processes. A lot of work has been done on security for single *steps* of such processes, e.g., “payments.” But still it is unclear how to compose a secure process out of secure steps, in a general way.

4. Beyond data encryption, not much technology is available that supports the users’ privacy and anonymity. Even where individual steps can be anonymized, like payments, it is not clear how to ensure anonymity across a whole business process (*pervasive anonymity*).

5. Public key infrastructures (PKI) are at the heart of many solutions but there is only little experience with large scale PKIs and only insufficient understanding of the semantics of certificates.

6. Many electronic commerce systems use a centralized trust model, not the more appropriate model of multi-party security. Often there are not even means for fair dispute handling.

7. Criteria and techniques for a formal evaluation of the security of electronic commerce systems are needed. Currently not even good models exist, not to speak about formal security proofs, for most systems.

8. There is no trustworthy computing base, i.e., there are no sufficiently secure, commercial operating systems or secure (mobile) user devices available.

9. The current user interfaces for security systems are by far too technology oriented. Metaphors and human oriented interfaces are needed.

Acknowledgments

Many thanks for discussions about the topics of this paper to *Tom Beiler, Gérard Lacoste, Thomas Hecht, Christian Petersen, Tomas Sander, Matthias Schunter, Arnd Weber,* and *Rita Zihlmann.* Special thanks to *N. Asokan, Birgit Baum-Waidner, Birgit Pfitzmann* and *Michael Steiner* who contributed many excellent ideas and comments.

References

[AABB97] Hal Abelson, Ross Anderson, Steven M. Bellovin, Josh Benaloh, Matt Blaze, Whitfield Diffie, John Gilmore, Peter G. Neumann, Ronald L. Rivest, Jeffery I. Schiller, Bruce Schneier: The Risk of Key Recovery, Key Escrow, and Trusted Third-Party Encryption; The World Wide Web Journal 2/3 (1997) 241-257.

[AASW98] Jose L. Abad-Peiro, N. Asokan, Michael Steiner, Michael Waidner: Designing a Generic Payment Service; IBM Systems Journal 37/1 (1998) 72–88.

[Abe98] Masayuki Abe: Universally Verifiable Mix-Net with Verification Work Independent of the Number of Mix-Servers; Eurocrypt ’98, LNCS 1403, Springer-Verlag, Berlin 1998, 437–447.

[ACPZ98] C. Adams, P. Cain, D. Pinkas, R. Zuccherato: Time Stamp Protocols; Internet Draft draft-adams-time-stamp-02.txt, June 4, 1998.

[AdZu97] C. Adams, R. Zuccherato: Notary Protocols; Internet Draft draft-adams-notary-01.txt, February 27, 1997.

[AJSW97] N. Asokan, Phillipe A. Janson, Michael Steiner, and Michael Waidner: The State of the Art in Electronic Payment Systems; Computer 30/9 (1997) 28-35.

[AnKu96] Ross Anderson, Markus Kuhn: Tamper Resistance – A Cautionary Note; 2nd USENIX Workshop on Electronic Commerce, 1996, 1–12.

[Asok98] N. Asokan: Fairness in Electronic Commerce; PhD thesis, University of Waterloo, May 1998.

[AsSW97] N. Asokan, Matthias Schunter, Michael Waidner: Optimistic Protocols for Fair Exchange; 4th ACM Conference on Computer and Communications Security, Zürich, April 1997, 6–17.

[AsSW98] N. Asokan, V. Shoup, Michael Waidner: Asynchronous Protocols for Optimistic Fair Exchange; 1998 IEEE Symposium on Research in Security and Privacy, IEEE Computer Society Press, Los Alamitos 1998, 86–99.

[AsSW198] N. Asokan. Victor Shoup, Michael Waidner: Optimistic Fair Exchange of Digital Signatures; Eurocrypt '98, LNCS 1403, Springer-Verlag, Berlin 1998, 591–606.

[AsVS98] N. Asokan, Els Van Herreweghen, Michael Steiner: Towards a Framework for Handling Disputes in Payment Systems; Third USENIX Workshop on Electronic Commerce, Boston, September 1998, 187–202.

[BaHS93] Dave Bayer, Stuart Haber, W. Scott Stornetta: Improving the efficiency and reliability of digital time-stamping; Sequences II: Methods in Communication, Security, and Computer Science, 329–334, New York, 1993. Springer-Verlag.

[Bako98] Yannis Bakos: The Emerging Role of Electronic Marketplaces on the Internet; Communications of the ACM 41/8 (1998) 35–42.

[BBCM194] Jean-Paul Boly, Antoon Bosselaers, Ronald Cramer, Rolf Michelsen, Stig Mjøls-nes, Frank Muller, Torben Pedersen, Birgit Pfitzmann, Peter de Rooij, Berry Schoenmakers, Matthias Schunter, Luc Vallée, Michael Waidner: The ESPRIT Project CAFE — High Security Digital Payment Systems; ESORICS 94 (Third European Symposium on Research in Computer Security), Brighton, LNCS 875, Springer-Verlag, Berlin 1994, 217–230.

[BGGM98] Daniel Bleichenbacher, Eran Gabber, Phillip B. Gibbons, Yossi Matias, Alain Mayer: On Secure and Pseudonymous Client-Relationships with Multiple Servers; 3rd USENIX Workshop on Electronic Commerce, Boston, Massachusetts, August 31 – September 3, 1998.

[BlFK98] Matt Blaze, Joan Feigenbaum, Angelos D. Leromytis: KeyNote: Trust Management for Public-Key Infrastructures; AT&T Research, 1998.

[BlFL196] Matt Blaze, Joan Feigenbaum, Jack Lacy: Decentralized Trust Management; 1996 IEEE Symposium on Security and Privacy, IEEE Computer Society Press, Washington 1996, 164-173.

[BLLW98] Ahto Buldas, Peeter Laud, Helger Lipmaa, Jan Villemson: Time-Stamping with Binary Linking Schemes; Crypto '98, LNCS 1462, Springer-Verlag, Berlin 1998, 486–501.

[BoDL97] Dan Boneh, Richard A. DeMillo, Richard J. Lipton: On the Importance of Checking Cryptographic Protocols for Faults; Eurocrypt '97, LNCS 1233, Springer-Verlag, Berlin 1997, 37–51.

[BuePf89] Holger Bürk, Andreas Pfitzmann: Digital Payment Systems Enabling Security and Unobservability; Computers & Security 8/5 (1989) 399–416.

[CFSY96] Ronald Cramer, Matthew Franklin, Berry Schoenmakers, Moti Yung: Multi-authority secret-ballot elections with linear work; Eurocrypt '96, LNCS 1070, Springer-Verlag, Berlin 1996, 72–83.

[DeBr96] Dorothy E. Denning, Dennis K. Branstad: A Taxonomy for Key Escrow Encryption Systems; Communications of the ACM 39/3 (1996) 34-39.

[DeRi98] Thomas Demuth, Andreas Rieke: Anonym im World Wide Web?; Datenschutz und Datensicherheit DuD 22/11 (1998) *to appear.*

[Chau81] David Chaum: Untraceable Electronic Mail, Return Addresses and Digital Pseudonyms; Communications of the ACM 24/2 (1981) 84–88.

[Cha885] David Chaum: Security without Identification: Transaction Systems to make Big Brother Obsolete; Communications of the ACM 28/10 (1985) 1030–1044.

[CrLa98] Lorrie Faith Cranor, Brian A. LaMacchia: Spam!; Communications of the ACM 41/8 (1998) 74–83.

[EaCS94] Donald E. Eastlake, Stephan D. Crocker, Jeffrey I. Schiller: Randomness Requirements for Security; IETF RFC1750, 1994.

[ECBS97] European Committee for Banking Standards: Secure Banking over the Internet; TR 401, European Committee for Banking Standards (ECBS), Avenue de Tervueren, 12, 1040 Brussels; March 1997. Edited by Rainer Rueppel.

[Feig98] Joan Feigenbaum: Towards an Infrastructure for Authorization (Position Paper); 3rd USENIX Workshop on Electronic Commerce, Boston, Massachusetts, August 31 – September 3, 1998 (not in printed version of proceedings).

[FoBa97] Warwick Ford, Michael S. Baum: Secure Electronic Commerce; Prentice Hall, Upper Saddle River 1997.

[Frit98] Lothar Fritsch: Sichere anonyme Kommunikation für den elektronischen Marktplatz *SEMPER* – Design und Implementierung in Java; Diplomarbeit, Fachbereich Informatik, Universität des Saarlandes, Saarbrücken, 7.8.1998.

[FrKK96] Alan O. Freier, Philip Karlton, Paul C. Kocher: The SSL Protocol Version 3.0; Internet-Draft, Transport Layer Security Working Group, 18.11.1996.

[FrRe95] Matthew K. Franklin, Michael K. Reiter: The Design and Implementation of a Secure Auction Service; 1995 IEEE Symposium on Research in Security and Privacy, IEEE Computer Society Press, Los Alamitos 1995, 2-14

[GGMM98] E. Gabber, P. Gibbons, Y. Matias, A. Mayer: How to Make Personalized Web Browsing Simple, Secure, and Anonymous; Financial Cryptography 97, February, 1997, Springer-Verlag, LNCS 1318.

[HaSt91] Stuart Haber, W. Scott Stornetta: How to Time-Stamp a Digital Document; Journal of Cryptology 3/2 (1991) 99-111.

[HuPf96] Michaela Huhn, Andreas Pfitzmann: Technische Randbedingungen jeder Kryptoregulierung; Datenschutz und Datensicherheit DuD 20/1 (1996) 23–26.

[Inte97] Intel Corp.: Common Data Security Architecture; 27.10.1997. `http://www.intel.com/ial/security/cdsa/index.htm`

[Jako98] Markus Jakobsson: A Practical Mix; Eurocrypt '98, LNCS 1403, Springer-Verlag, Berlin 1998, 448–461.

[JECF98] JavaSoft: Electronic Commerce; 1998. `http://java.sun.com/commerce`

[JMPP98] Anja Jerichow, Jan Müller, Andreas Pfitzmann, Birgit Pfitzmann, Michael Waidner: Real-Time Mixes: A Bandwidth-Efficient Anonymity Protocol; IEEE Journal on Selected Areas in Communications 16/4 (1998) 495–509.

[Koch96] Paul Kocher: Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems; Crypto '96, LNCS 1109, Springer-Verlag, Berlin 1996, 104–113.

[Maur196] Ueli Maurer: Modelling a Public-Key Infrastructure; ESORICS '96 (4th European Symposium on Research in Computer Security), Rome, LNCS 1146, Springer-Verlag, Berlin 1996, 325–350.

[OBI98] The OBI Consortium: Open Buying on the Internet (OBI) – Technical Specifications, Release V1.1; The OBI Consortium, June 1998. `http://www.openbuy.org`

[OTP98] The OTP Consortium: Internet Open Trading Protocol (Version: 0.9.9); Open Trading Protocol Consortium, 17 August 1998. `http://www.otp.org`

[PfWa87] Andreas Pfitzmann, Michael Waidner: Networks without user observability; Computers & Security 6/2 (1987) 158–166.

[PfWa96] Birgit Pfitzmann, Michael Waidner: Properties of Payment Systems: General Definition Sketch and Classification; IBM Research Report RZ 2823 (#90126) 05/06/96, IBM Research Division, Zurich, May 1996.

[PPSW97] Andreas Pfitzmann, Birgit Pfitzmann, Matthias Schunter, Michael Waidner: Trusting Mobile User Devices and Security Modules; IEEE COMPUTER 20/2 (1997) 61–68.

[PWP87] Birgit Pfitzmann, Michael Waidner, Andreas Pfitzmann: Rechtssicherheit trotz Anonymität in offenen digitalen Systemen; Computer und Recht 3/10,11,12 (1987) 712–717, 796–803, 898–904; Extended revision in Datenschutz und Datensicherung DuD 14/5–6 (1990) 243–253, 305–315. (English translation available from authors.)

[ReRu97] Michael K. Reiter, Aviel D. Rubin: Crowds: Anonymity for Web Transactions; DIMACS Technical Report 97–15, April 1997, revised August 1997.

[ReSG98] Michael G. Reed, Paul F. Syverson, David M. Goldschlag: Anonymous Connections and Onion Routing; IEEE Journal on Selected Areas in Communications 16/4 (1998) 482–494.

[RiLa96] Ronald L. Rivest, Butler Lampson: SDSI — A Simple Distributed Security Infrastructure; Manuscript, April 30, 1996. `http://theory.lcs.mit.edu/~rivest/sdsi10.ps`

[SEMP1098] *SEMPER* Consortium: Architecture, Services and Protocols; Deliverable D10 of ACTS Project AC026, Public, December 31, 1998.

[SEMP1398] *SEMPER* Consortium: Final Report of Project *SEMPER*; Deliverable D13 of ACTS Project AC026, Public, December 31, 1998.

[SET97] Mastercard, Visa: Secure Electronic Transactions (SET) Version 1.0 – Book 1: Business Descriptions, Book 2: Programmer's Guide, Book 3: Formal Protocol Specification; Report, May 31, 1997.

[Turb97] Efraim Turban: Auctions and Bidding on the Internet: An Assessment; Intern. J. of Electronic Markets 7/4 (1997) 7–11. `http://www.electronicmarkets.org`

[Waid96] Michael Waidner: Development of a Secure Electronic Marketplace for Europe; ESORICS '96 (4th European Symposium on Research in Computer Security), Rome, LNCS 1146, Springer-Verlag, Berlin 1996, 1–14.

[Webe97] Arnd Weber: Zur Notwendigkeit sicherer Implementation digitaler Signaturen in offenen Systemen; in: Günter Müller, Andreas Pfitzmann (eds.): Mehrseitige Sicherheit in der Kommunikationstechnik, Addison-Wesley-Longman 1997, 465–478.

[Zeko98] Robert Hobbes Zakon: Hobbes' Internet Timeline v3.3; IETF RFC 2235, 1998. `http://www.isoc.org/guest/zakon/Internet/History/HIT.html`