e-Voting.at:
Entwicklung eines Internet-basierten Wahlsystems für öffentliche Wahlen

Alexander Prosser
Robert Kofler
Robert Krimmer
Martin Karl Unger

Arbeitspapiere zum Tätigkeitsfeld
Informationsverarbeitung und Informationswirtschaft
Working Papers on
Information Processing and Information Management

Nr./No. 04/2002

Herausgeber / Editor:
Institut für Informationsverarbeitung und Informationswirtschaft
Wirtschaftsuniversität Wien · Augasse 2-6 · 1090 Wien
Institute of Information Processing and Information Management
Vienna University of Economics and Business Administration
Augasse 2-6 · 1090 Vienna

pm production management

E-Mail: e-Voting@wu-wien.ac.at
WWW: http://www.e-Voting.at

Inhaltsverzeichnis

1 Management Summary

1.1 Deutsch

Internetwahlen (e-Voting) sind zu einer realen Möglichkeit geworden, es müssen aber die allgemeinen Wahlrechtsgrundsätze eingehalten werden. Bei der Entwicklung eines e-Voting-Systems müssen insbesondere folgende Probleme gelöst werden:

- eindeutige Identifikation des Wahlberechtigten bei der Registration für die elektronische Wahl bei gleichzeitig
- vollkommen gesicherter Anonymität in der Stimmabgabe.
- Außerdem darf die Systemadministration der Wahlbetreiber keinerlei Möglichkeit haben (i) die Anonymität zu unterlaufen oder (ii) Stimmen zu manipulieren.

Der vorliegende Prototyp basiert auf einem an der Abteilung Produktionsmanagement der WU Wien entwickelten Verfahren, das international publiziert und damit der öffentlichen Diskussion und Prüfung zugänglich ist (siehe dazu die Auswahl an Publikationen in diesem Bericht). Zur absoluten Sicherung der Anonymität teilt das Verfahren die Wahl in

- die Registrierungsphase, bei der sich der Wahlberechtigte identifiziert und die Ausstellung einer elektronischen Briefwahlkarte beantragt, und
- die Stimmabgabephase, bei der die elektronische Briefwahlkarte für die anonyme Stimmabgabe eingesetzt wird.

Zwischen diesen beiden Phasen wird die Briefwahlkarte auf einer Chipkarte (in Österreich sinnvollerweise der Bürgerkarte) zwischengespeichert.

Der vorliegende Prototyp realisiert den ersten Teil des Verfahrens, die Registrierung bei der elektronischen Wählerevidenz und das Ausstellen einer elektronischen Briefwahlkarte. Dabei wird die reale Infrastruktur der Bürgerkarte und des Zentralen Melderegisters (ZMR) sowie des Trust Center-Servers der Fa. Datakom eingesetzt:

- Identifikation des Wählenden über das ZMR
- Personenbindung der österreichischen Bürgerkarte (verknüpft das digitale Zertifikat im Trust Center mit der ZMR-Zahl der realen Person)
- digitale Unterschrift unter Nutzung des Security Layers der Bürgerkarte

Damit nutzt das System die reale Infrastruktur der Bürgerkarte und des ZMR.

Der Ablauf ist in 2.1 detailliert dargestellt, hier eine inhaltliche Zusammenfassung:

1. Der Wahlberechtigte ruft die Web-Applikation zur Registration auf.
2. Der Wahlberechtigte sendet einen unterschriebenen Antrag auf Ausstellung einer elektronischen Wahlkarte an den Registrator, der den Wahlberechtigten und seinen Wahlsprengel anhand der ZMR-Zahl identifiziert. Kann der Wahlberechtigte authentifiziert werden, sendet der Registrator eine blind signierte elektronische Briefwahlkarte zurück (zur blinden Signatur siehe weiter unten).
3. Der Wahlberechtigte sendet einen unterschriebenen Antrag auf Ausstellung einer elektronischen Prüfkarte an das Trust Center, das eine blind signierte Prüfkarte zurücksendet.
4. Wahl- und Prüfkarte werden beim Wahlberechtigten gespeichert.

Die blinde digitale Signatur ist wie die „normale“ digitale Signatur fälschungssicher, der Unterschreibende sieht aber nicht, was er unterschreibt. In der realen Erfahrung wäre dies mit der Unterschrift auf einem Kuvert aus Blaupapier vergleichbar, in dem eine Wahlkarte

liegt: die Unterschrift paust sich auf die Wahlkarte durch (ist daher fälschungssicher wie eine echte Unterschrift), der Unterschreibende sieht aber nur das Blaupapier, nicht die Wahlkarte) .

Zur Implementierung eines vollständigen Prototypen ist noch die zweite Phase des Verfahrens notwendig: das Einsetzen der elektronischen Wahlkarte für die anonyme Stimmabgabe. Dies ist Gegenstand eines laufenden Forschungsantrages beim Jubiläumsfonds der Stadt Wien.

Wir danken dem Jubiläumsfonds der Stadt Wien und der Wirtschaftsuniversität Wien für die finanzielle Unterstützung beim Bau dieses Prototypen sowie der Fa. Datakom GesmbH für die Bereitstellung digitaler Signaturkarten.

Wien, im November 2002

Alexander Prosser, Robert Kofler,
Robert Krimmer und Martin Unger

1.2 English

Internet voting (e-Voting) has become a real possibility, however, the general voting principles have to be adhered to also in internet voting. The following issues have to be addressed:

- eligible voters have to be identified when registering for e-voting,
- at the same time, anonymity in casting votes has to be ensured.
- Also, the administration of the election servers must not have any possibility to manipulate votes.

The prototype presented in this research report is based upon a procedure developed at the Dept. of Production Management at WU Vienna, which has been internationally published and which is therefore open to discussion and scrutiny (see the selection of papers in this report). To absolutely ensure anonymity, the process is divided into two parts:

- registration, where eligible voters may retrieve an electronic election token and
- voting, where the token is used to anonymously cast a vote.

This two-fold process requires the token to be stored on a secure medium, ideally a signature (smart) card, in Austria, this would be the Citizen Card (Bürgerkarte). The prototype implements the first part of the process, that is registration at the electronic voters' register and retrieval of an electronic election token. The prototype is embedded in the real infrastructure of the digital signature card (Bürgerkarte), Austria's Central Citizens Register (ZMR) and the Trust Center operated by Datakom Vienna:

- Voter identification using ZMR
- Using the link between the digital certificate provided by the Trust Center and the citizen's ZMR ID; this link is the decisive feature of Austria's Bürgerkarte
- Digital signature using the security layer provided by the Bürgerkarte

The process is detailed in 2.1, the following itemization provides a brief high-level overview:

1. The eligible voter retrieves the Web application for electronic voter registration.
2. The voter sends a digitally signed application for an electronic election token to the Registrator, which in turn authenticates the voter using the ZMR ID provided and determines his/her constituency. If the voter can be authenticated the Registrator returns a signed election token. Since blind signature is used (see explanation below), the Registrator does not "see" the token it signs, which preserves voter anonymity.
3. A similar protocol is executed with the Trust Center, where the voter obtains a second blindly signed token, which is used for additional safety.
4. Both tokens are stored at the voter.

Similar to „normal“ digital signatures, blind signatures provide a secure medium of authentication; in this case, however, the signor does not “see” the message which it signs. To draw on a real-world example, a blind signature would correspond to signing a document wrapped in a carbon paper envelope: The signature traced on the document is genuine, but the signor never sees the document itself. Blind signatures are also used for digital cash.

The prototype presented herein, implements the registration phase. To implement the entire protocol, the second stage is still to be implemented: to use the electronic token to cast a vote. An application for a research grant is currently under review at the Anniversary Foundation of the City of Vienna.

We gratefully acknowledge the support of our research work by the Anniversary Foundation of the City of Vienna and the University of Economcis and Business Administration Vienna. We owe thanks to Datakom Vienna for providing digital signature cards.

Vienna, in November 2002

Alexander Prosser, Robert Kofler,
Robert Krimmer and Martin Unger

2 Entwicklung einer e-Voting-fähigen Wählerevidenz

2.1 Implementation Report

Abstract: [1] *Worldwide research groups have developed remote electronic voting systems using several different approaches with no legal basis. In 2001 the Austrian Parliament passed a law allowing electronic voting with digital signatures for public elections. Besides these legal requirements, an algorithm has to solve the basic technical problem, of how to identify the user uniquely with still guaranteeing the anonymity of one's vote and further not to allow fraud by the election administration.*
In this paper the authors give an experience report on the implementation of the first phase of an algorithm that fulfills these requirements by strictly separating the registration from the vote submission phase.

1 INTRODUCTION

Worldwide sinking voter turn out has led to various electronic voting research projects in the field of remote electronic voting to allow secure and legally binding public elections over the Internet. With elections being the key element of a democratic system, the way they are held is determined by the political traditions, the social context and the legal system. However, most systems know the principles of free, equal, and secret elections, which result in the following basic problem to solve for every electronic voting system:

- the voter must be identified uniquely, but must still be able to
- cast her vote anonymously and
- even the administrators must not be able to change this vote

In Austria the voter turnout of nation-wide elections has been constantly above 90% until 1994 when it first dropped below that mark and reached an all time low of 80,4 % in 1999 [1]. While this number is still high by international standards, first initiatives in favor of introducing means of distance (remote) voting for public elections came up [2]. So far only Austrians abroad are entitled to vote by mail in first-order elections, this being elections to the Parliament and to the Federal President [for further elaboration see 3].

In 2001 the Austrian Parliament passed two bills allowing electronic voting for elections of two organizations: the student union and the chamber of commerce. The laws also require the electronic voting system to use digital signatures and to be approved by Austria's data protection commission and by the national IT certification organization A-SIT.

In a research project funded by the City of Vienna a research group develops an electronic voting prototype in two development stages: First an electronic voting enabled registration is developed and based on this an electronic urn completes the prototype.

In this paper the authors first describe the algorithm used and then describe in a process model how the concept was implemented and which extensions were necessary due to further developments in the field of e-government. One example would be the concept of the Austrian national ID card that has not been available at the time the original algorithm was developed.

2 ALGORITHM

The algorithm under discussion was first proposed in [4] and then further developed in the preparation of the implementation project. The basic characteristic of it is the strictly separation of registration and vote submission stage first identified by Nurmi et.al. in 1991 [5]:

[1] This article was written by Prosser, Kofler and Krimmer with the title "Implementing an Internet-based Voting System for Public Elections – A Project Experience" and submitted to the International Enterprise Information Systems Conference 2003.

1. Registration phase: The voter's credentials are checked and the voter receives a blindly signed voting card, which is securely stored.

2. Vote submission phase: The voter uses the voting card to obtain a ballot sheet and casts her vote.

Before we describe the process model, let us begin with some notation:

- RS Registration Server
  TC Trust Center
  US Urn (Ballot Box) Server
  RegDB Registration Database

- pi Voter's personal identification file
  c Voter's constituency

- e, d Registration's public and private signature key
  k, l Registration's public and private crypto key

- ε, δ Trust Center's public and private signature key
  κ, λ Trust Center's public and private crypto key

- u, v Voter's public and private signature key
  w, z Voter's public and private crypto key

- υ, ϖ Urn's public and private signature key
  ω, ζ Urn's public and private crypto key

- r, t random numbers for voting card
  ρ, τ random numbers for validation card
  x, ξ blinded Voting and Validation card

- m, m' random asymmetric crypto key pair for voting process

2.1 Registration

Voters can register an arbitrary period of time before election day; since the ballot sheet is not handed out upon registration, voters can register even at a time when the list of candidates is not complete yet. As the first step the voter generates the random numbers for the voting card r, t and prepares it for the blind signature[2], adds a text where she applies for electronic voting and signs: v(x, "I want to e-vote"). The message is encrypted with registration's public crypto key k and sent to the registration: k[v(x, "I want to e-vote")], which verifies the voter's credentials by resolving the public signature key of the voter. If the voter is entitled to vote, the registration signs x blindly giving x^d. After receiving x^d the voter can access the signed voting card t^d by dividing x^d by r.

The registration stores the electronic application and strikes the voter off the conventional voter's register. Also x^d is stored; if the original signed card is lost and the voter re-applies for another voting card, the registration will always respond with the original x^d to avoid the issue of multiple cards.

In most elections, voters will be organized in constituencies c, this information is also sent back to the voter and has to be submitted on election day to indicate in which constituency the vote is to be counted. To avoid possible manipulation of c the blind signature keys used for x^d can be made specific to the constituency. Hence the clear-text c submitted on election day and the authentication card issued by the registration have to point to the same c.

---

[2] The blind signature model was developed by David Chaum in 1982 [6]. In general language it can be compared with the signature on a blue paper envelope.
(e, d) ... the server's blind signature pair
x = re*t
xd = (re*t)d; which is then divided by r
giving [(re)d *td] / r = td

A similar process is repeated with the Trust Center: The voter issues a second pair of random numbers ρ, τ and then prepares the blinded validation card ξ and obtains the blindly signed ξ^{δ}. By dividing again ξ^{δ} by ρ she gains the signed validation card τ^{δ}. This is required as it is the only way to make a collusion of the registration server and the ballot box server useless, as they always need the blind signature authentication of the Trust Center as well in order to forge a vote.

At the end of the registration phase, the voter holds two authentication cards and her constituency information [t, t^d, τ, τ^{δ}, c], both voting and validation card are needed to cast a vote on election day.

2.2 Voting

On election day the voter sends the cards to the ballot box server to obtain a ballot sheet. This submission is not signed by the voter and the only means of authentication are the two cards obtained earlier. The voter generates an asymmetric key *m*, m' to secure the communication (without disclosing the identity of the voters which would be case when using its crypto key pair on the signature smart card). The voter also adds the identification TC of the Trust Center used, which is not used to verify the voter's identity or to obtain any public crypto key, but to choose the right Trust Center key for resolving the blind signature. The message [TC, m, t, t^d, τ, τ^{δ}, c] is encrypted with urn's public crypto key ω and sent to the ballot box. After decrypting the ballot box resolves the signatures t^d and τ^{δ} and if the cards can be authenticated the ballot box issues an empty ballot sheet and encodes it with the symmetric key m(BS). The voter receives and decrypts it with m' and fills out the ballot sheet. This is then combined with the cards, c and TC, encrypted again with the public crypto key of the ballot box and sent. After authentication of the cards, the ballot box server stores the ballot sheet and the other information received from the voter.

Apart from the fact that anonymity can be guaranteed to the voter, if he uses different terminals (IP addresses) for registration and submission of vote, the server administration of the registration and the ballot box collude, votes cannot be forged, as a valid vote also has to be authenticated by a Trust Center.

3 IMPLEMENTATION

In Austria the concepts of e-Government and e-Administration have become an issue of high importance on the politicians' agenda. This resulted in the installation of the CIO (chief information office). The main task is to develop a coordinated strategy for e-Government in the Austrian administration [7].

This strategy concentrates on issues of infrastructure and this makes the introduction of a national ID card based on the digital signature law an issue of high importance.

As described in the introduction, the digital signature on a smart card is one of the requirements for electronic voting by Austrian law[3]. When using smart cards to sign digital documents, everyone can access a public certificate server to verify the validity of the digital signature. But when one wants to uniquely identify a user, this is concept is not enough. Even if the user credentials comply with the person in the election register, one can not be sure that they are identical because only the name and date of birth are not enough to uniquely identify one user. This problem can be solved if either (i) the smart card issuing organization is the same as the election conducting body or if (ii) one uses a unique identifier like a citizen ID number that is stored in the personal identification file *pi* on the smart card.

While the first solution is useful for elections that have full control over the ID cards their voters use, it is not

[3] The first larger number of smart cards being rolled out is the student ID card of the Vienna University of Economics and Business Administration (WU) [8]

useable for elections where the user may choose freely which ID card she may use. The second variant is a topic of infrastructure, as it requires a central register of all citizens. This central database not only involves problems of data acquisition and maintenance but also of data protection. That is why it is in most countries prohibited by law or by constitutional court like in Germany. Surprisingly in Austria this is not the case and the registration law 1995 installed such a central register (ZMR). It started public service on first of March 2001 and every Austrian citizen has been appointed one unique "ZMR-identification-number". However, in spite of its usefulness for electronic voting, it should be noted that critics like the data security specialists from ARGEdaten warned from misuse and called the system the first step to a surveillance state [9].

With this ZMR-identification number stored in the personal identification file *pi* on the citizen's smart card it becomes a national ID card. In this way one can use it for the identification process for checking the citizen's eligibility to vote and the issuing of the voting card.

One further very useful technical development atop of the nation ID card, the chief information office developed a security layer for use with this card [10]. Basically it is a standard application interface for signing and verification purposes in form of local http server on the user's computer. It handles all interactions between the users Internet browser and the smart card by using standard HTML forms and makes therefore interaction with the smart card possible without Java.

These recent developments have now been implemented in the electronic voting prototype. This has led to a new five step process within the first stage of the protocol, which is now described in detail:

1. Applet download
2. Validation card Preparation
3. Authorization check and voting card preparation
4. Blind signature of the validation card
5. Blind signature of the voting card

3.1 Step One: Applet Download

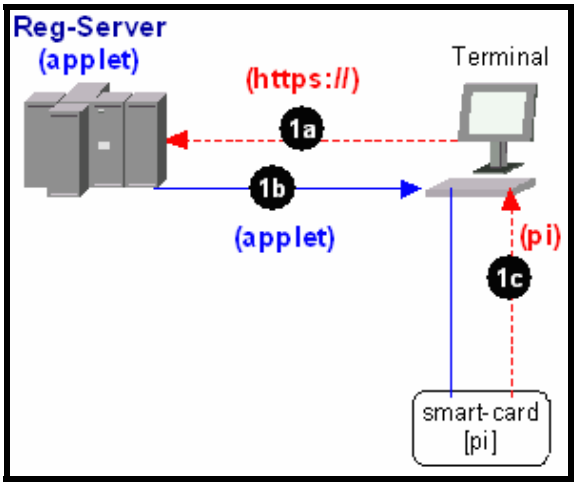


Figure 1: Download of the registration application

In the beginning the voter starts the remote electronic voting process by entering the URL provided by the election administration (1a). Then the Web client of the voter downloads the registration java applet from the registration server (1b). The first action is the applet reading in (1c) the personal identification file pi from the smart card (via a procedure call of the security layer) in preparation of the authorization check in step 3.

3.2 Step Two: Validation card Preparation

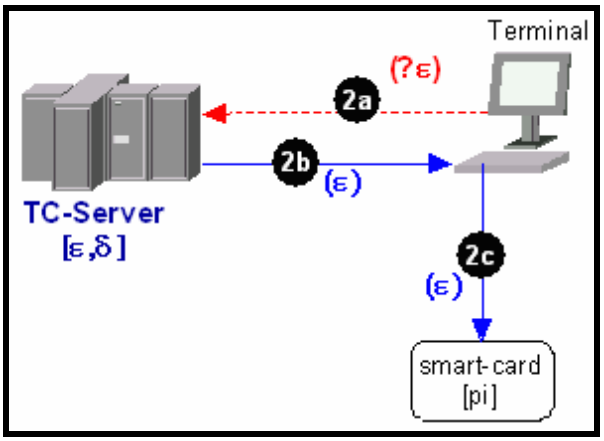


Figure 2: Preparation of the validation card

For the blinding process one takes the random number r and signs it with the public signature key of the server resulting in r^e. Afterwards the applet requests in (2a) this public key from the Trust Center, receives it in (2b) and stores it on the smart card for the blinding of the validation card later on (2c).

3.3 Step Three: Authorization check and voting card preparation

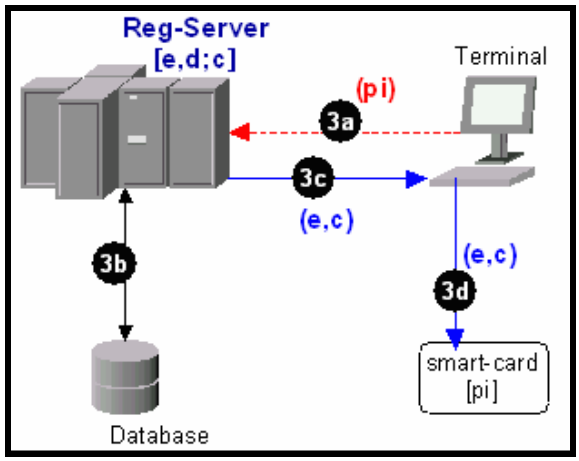


Figure 3: Authorization check/voting card preparation

The third step is basically a repetition of step two but just with the registration server with a check of the voter's eligibility. Hence, the check of the citizen's authorization is also checked here, so the voter can be correctly assigned a constituency c. To facilitate that, the applet sends the previously loaded pi to the registration server and requests the public key (3a). Using the registration database the server determines the citizen's authorization to vote and her constituency (3b). Following the reception of e and c (3c) the applet stores it on the smart card for the blinding of the voting card in (3d).

3.4 Step Four: Blind signature of the validation card

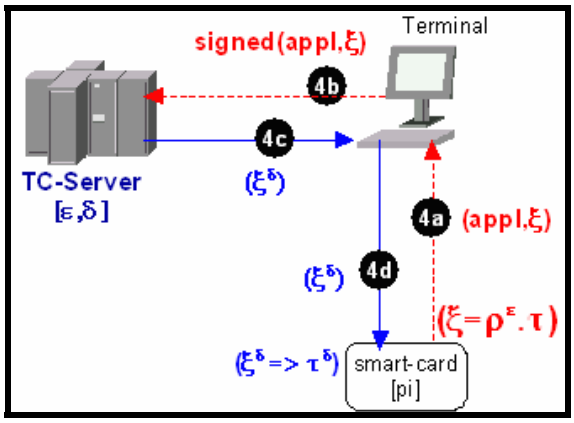


Figure 4: Blind signature of the validation card

The validation card is being blinded in (4a) by taking the random number ρ and signing it with ε resulting in the blinded card: $\xi = (\rho^{\varepsilon}\tau)$ and then sending it with an application ("I want to vote electronically") in (4b) to the Trust Center. This is then signed giving ξ^{δ} and sent back to the voter. In (4d) the voter stores the package on the smart card where it is then processed to receive the final validation card t^d (4e).

3.5 Step Five: Blind signature of the voting card

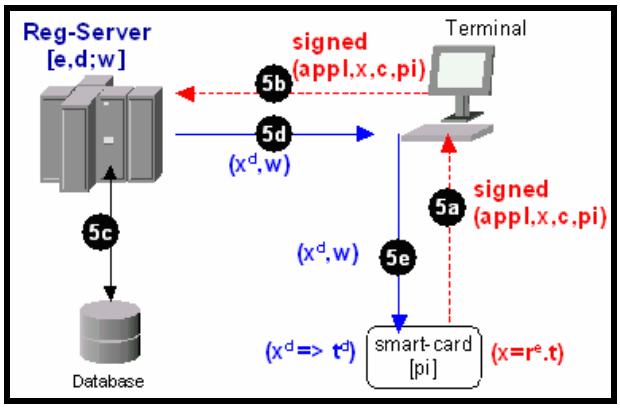


Figure 5: Blind Signature of the voting card

As a final process the voter is applying for the voting card by first retrieving the blinded voting card x that consists out of (r^e * t) from the smart card (5a). It is then followed by sending it together personal identification file, the constituency and an application like "I want to vote electronically" to the registration server (5b). The registration is then again looking up the citizen's authorization to vote and checking the constituency using the personal identification (5c). If this is validated the registration server signs the blinded voting card. Further the server sends x^d with the constituency back to the voter and in case of success marks the citizen with a "has received voting card" tag (5d). In the following final process (5e) x^d is stored in the smart card where t^d is gained by dividing by r and stored together with c.

4 RESUME

The paper describes a protocol for public elections using remote electronic voting over the Internet. It not only solves the problem of casting a vote anonymously but also works under unfriendly conditions like when the registration server's and the urn server's administration are colluding.

But as election procedures vary from country to country, the implementation of such a protocol always has to adapt to local specialities and recent developments.

In the case of Austria two developments assist the project – on one side the clear legal requirements for an electronic voting system beforehand and on the other side the work of the Chief Information Office on the national ID card with concepts like the security layer.

The further work of the authors concentrates on the implementation of the second phase of the prototype, so the algorithm can be tested in a real world (mock) election to gain further experience on how to improve it.

5 REFERENCES

[1] "Die österreichischen Nationalratswahlen von einst bis heute", http://www.modernpolitics.at/publikationen/jahrbuch/wahlergebinsse/wahlen_index.htm, as available on 2002-09-25

[2] J. Weiss, Gesetzesantrag "Einführung der Briefwahl auf Landes- und Gemeindeebene", http://www.parlinkom.at/pd/pm/XXI/I/his/000/I00005_.html, as available on 2002-08-15

[3] W. Dujmovits, *Auslandsösterreicherwahlrecht und Briefwahl*. Wien: Verlag Österreich, 2000.

[4] A. Prosser and R. Müller-Török, "Electronic Voting via the Internet," presented at International Conference on Enterprise Information Systems ICEIS2001, Setùbal, 2001.

[5] H. Nurmi, A. Salomaa, and L. Santean, "Secret Ballot Elections in Computer Networks," *Computers and Security 36 (10)*, pp. 553-560, 1991.

[6] D. Chaum, "Blind Signatures for Untraceable Payments," presented at Advances in Cryptology, 1982.

[7] e-Austria in e-Europe, http://www.austria.gv.at/aktuell/database/topnews/german/20000414_713.html, as available on 2002-08-15

[8] Wirtschaftsuniversität Wien, http://www.wu-wien.ac.at, as available on 2002-08-15

[9] Überflüssiges Meldegesetz, http://www.ad.or.at/news/20010108.html, as available on 2002-08-15

[10] A. Hollosi and G. Karlinger, "Security-Layer für das Konzept Bürgerkarte," BMÖLS, CIO Unit, Wien 2002.

2.2 Installationsanleitung

Das Applet ElvisCup macht innerhalb des Projektes den Teil Electronic Voter Registration. In dieser Einführung erfahren sie, wie man die Software für den Client, also für den PC des Wählers, installiert und startet. [4]

Installation der PC-Software

Das Applet ElvisCup braucht das Java Runtime Environment (JRE) oder das Java Software Development Kit (Java SDK) ab Version 1.4.0. Um festzustellen, ob Java auf Ihrem PC bereits installiert ist, geben Sie auf der Kommandozeile den Befehl java -version ein. Wenn die Version 1.4.0 noch nicht installiert ist, dann können Sie die Datei j2sdk-1_4_0-win.exe von der CD-ROM auf die Festplatte kopieren und durch Doppelklick entpacken und installieren. Auf der Web-Site der Firma Sun (http://java.sun.com/) kann man bereits die Version 1.4.1 finden.

Installation vom Security Layer

Der Security Layer ist eine Schnittstelle für die Bürgerkartenumgebung. Die SecurityKapsel als Implementation dieser Schnittstelle macht die für ElvisCup notwendigen digitalen Signaturen. Kopieren Sie das Verzeichnis security_layer auf Ihre Festplatte.

Installation vom Kartenstapel

Der Kartenstapel ist ein Verzeichnis, dessen Unterverzeichnisse die Signaturkarten symbolisieren. Kopieren Sie dieses Verzeichnis auf die Festplatte. Die html-Dateien, welche das Applet aufrufen, übergeben als Ausgangspunkt für die Suche nach dem Kartenstapel den String c:\kartenstapel\ weshalb es von Vorteil ist, wenn der Kartenstapel genau dort zu finden ist.

Installation des Applets (optional):

Wenn Sie das Applet nicht vom Server laden sondern von der lokalen Festplatte starten wollen, dann kopieren Sie folgende Dateien auf Ihre Festplatte:

- ElvisCup.jar
- ElvisCup.htm
- ElvisCupDebug.htm
- ElvisCupSeven.htm

Die erstgenannte Datei ElvisCup.jar enthält das kompilierte und signierte Java-Programm ElvisCup. Die anderen genannten Dateien sind .html-Dateien welche dazu dienen, im Web-Browser Microsoft Internet Explorer (ab Version 5.0) das Java-Plug-In zu starten sowie das Applet ElvisCup zu laden und auszuführen. Diese .html-Dateien enthalten unter anderem auch die absoluten Adressen (Uniform Resource Identifier) der drei PHP-Programme, mit denen das Applet ElvisCup kommuniziert.

Benötigte Browser-Software

Die drei html-Dateien ElvisCup.htm, ElvisCupDebug.htm, ElvisCupSeven.htm benötigen als Browser-Software den Microsoft Internet Explorer ab Version 5.0, damit das Java-Plug-In geladen wird.

Wenn Sie diese .html-Dateien umwandeln wollen, damit Sie das Applet ElvisCup auch mit dem Web-Browser Netscape Navigator verwenden können, dann lesen Sie bitte an der Adresse http://java.sun.com/j2se/1.4.1/docs/guide/plugin/developer_guide/html_converter_more.html Informationen über den HTML-Konverter HtmlConverter.exe welcher im Lieferumfang des Java SDK 1.4.0 enthalten ist.

[4] Das Applet ElvisCup wurde von Martin Karl Unger programmiert.

Starten des Applets ElvisCup

1. Starten der SecurityKapsel:
   Starten Sie im Verzeichnis security_layer die Stapeldatei SecurityKapsel.bat um die Security-Kapsel als lokalen Web-Server zu starten. Dieses Programm wird die beiden digitalen Signaturen erzeugen.

2. PIN notieren:
   Holen Sie sich aus der Datei card.properties im Verzeichnis security_layer\data-ecard den PIN für die digitalen Signaturen. Er lautet: iaik!ecdsa . Das Echt-System wird mit echten Chipkarten arbeiten und voraussetzen, dass jeder Inhaber einer solchen Chipkarte den PIN seiner Chipkarte kennt.

3. Browser starten:
   Starten Sie die Browser-Software.

4. Applet laden und starten:
   Laden sie eine der drei Dateien ElvisCup.htm, ElvisCupDebug.htm, ElvisCupSeven.htm entweder vom Server oder von der lokalen Festplatte, damit das Applet aus der Datei ElvisCup.jar geladen wird.

Benutzerinteraktion

Ein Mausklick auf die linke, mit Personenbindung laden beschriftete Schaltfläche startet das Applet. Es erscheint ein File-Dialog zum Suchen der Personenbindungsdatei. In jedem Unterverzeichnis des Kartenstapels ist ein Personenbindungsdatei. Nehmen Sie eine .XML-Datei, deren Name mit Personenbindung beginnt.

Wenn Sie das Applet ElvisCup mittels der .html-Datei ElvisCup.htm gestartet haben, dann werden die weiteren sechs Schritte automatisch gestartet; wenn Sie eine der beiden anderen .html-Dateien (ElvisCupSeven.htm/ElvisCupDebug.htm) verwendet haben, dann wird jeder der sechs folgenden Schritte ebenfalls mithilfe der linken Schaltfläche gestartet.

Die Bürgerkartenumgebung Security Layer wird Sie zweimal zur PIN-Eingabe auffordern. Bei diesen Gelegenheiten können Sie den Text, den Sie digital signieren, betrachten.

Am Ende der Verarbeitung kommt wieder ein File-Dialog zum Speichern der soeben erstellten elektronischen Wahlkarte. Das Applet ElvisCup schlägt Ihnen zum Speichern das Verzeichnis vor, aus dem Sie die Personenbindungsdatei geladen haben. (Dieses Verzeichnis symbolisiert eine Chipkarte.) Als Dateiname wird das Applet ElvisCup Ihnen den Namen waka.htm vorschlagen. Diese Datei können Sie dann mit einem Web-Browser betrachten.

Die elektronische Wahlkarte (in der Datei waka.htm)

Das Feld radix ist die Basis des Zahlensystems der nachfolgenden Zahlen. Der Wert 16 bedeutet, dass die nachfolgenden Zahlen im Hexadezimalsystem dargestellt sind. Die beiden Werte tokenT und tokenT2 sind vom Applet ElvisCup erstellte Zufallszahlen, die beiden Werte tokenTd und tokenT2d2 sind die im Zuge der blinden Signatur erstellten digitalen Unterschriften, welche mit dem private key des Wahlsprengels beziehungsweise dem private key des Trust Center Blind Signature Servers erstellt wurden.

Bitte beachten Sie, dass der Server die Tatsache der Ausstellung einer elektronischen Wahlkarte speichert und dass jeder Bürger pro Wahl nur maximal eine elektronische Wahlkarte erhält, weil er pro Wahl auch nur eine Stimme hat. Wenn Sie dieselbe Personenbindungsdatei ein zweites mal zur Ausstellung einer elektronischen Wahlkarte verwenden, ohne vorher die Datenbank auf dem Server zu editieren, dann erhalten Sie eine entsprechende Meldung, die besagt, dass der Bürger bereits eine elektronische Wahlkarte für diese Wahl erhalten hat.

2.3 Die Mathematik des Applet ElvisCup

1 EINLEITUNG[5]

Das Applet erzeugt gemeinsam mit dem Server, also dem Computer der Wahlbehörde, eine elektronische Wahlkarte. Die elektronische Wahlkarte ist ein Datenbestand, der zwei Eigenschaften erfüllt:

Anhand dieses Datenbestandes läßt sich beweisen, dass der Server an der Erstellung dieses Datenbestandes mitgewirkt hat. Damit ist sichergestellt, dass ausschließlich wahlberechtigte Bürger eine elektronische Wahlkarte erhalten.

Dieser Datenbestand läßt keine Rückschlüsse auf die Identität des Wählers zu. Damit bleibt das Wahlgeheimnis gewahrt.

Dazu wird ein als blind signature bezeichneter Mechanismus verwendet. Diese englische Bezeichnung läßt sich als "Leistung einer Unterschrift auf einen verdeckten Text" übersetzen. Verwendet wird dazu das im vorliegenden Text erläuterte RSA-Verfahren der public-key-Kryptographie.

2 DIE PUBLIC-KEY-KRYPTOGRAPHIE

Bei der Datenkommunikation mithilfe der public-key-Kryptographie verwendet jeder Kommunikationsteilnehmer ein aus zwei Schlüsseln bestehendes Schlüsselpaar: der sogenannte public key ist öffentlich bekannt, während der sogenannte private key nur seinem Eigentümer bekannt ist.

Verschlüsselung: Bevor man einen Text verschickt, verschlüsselt man diesen Text mit dem *public key* des Adressaten. Sobald der Adressat den verschlüsselten Text erhalten hat, kann er ihn mit seinem *private key* entschlüsseln.

Digitale Signatur: Der Kommunikationspartner, der die Unterschrift leistet, verschlüsselt den Text (oder eine auf der Grundlage dieses Textes berechnete Zahl, die als Hashwert bezeichnet wird) mit seinem eigenen *private key*. Jeder Empfänger des Textes kann dann mit dem *public key* des Unterzeichners feststellen, dass der Text und die digitale Signatur zusammenpassen.

Blinde Signatur: Bei der *blind signature*, also der Unterschrift unter einen verdeckten Text, erstellt der erste Kommunikationspartner (zum Beispiel Alice) einen Text und verschlüsselt ihn (mit einem nur ihr bekannten Schlüssel) und gibt ihn weiter an den Unterzeichner (zum Beispiel Bob). Bob erstellt eine digitale Signatur indem er den erhaltenen Text (den er nicht lesen kann) mit seinem *private key* verschlüsselt. Alice kann dann mit einfachen mathematischen Umformungen Bobs digitale Signatur für den ursprünglichen Text erstellen. Dazu benötigt sie den ursprünglichen Text, den von ihr verwendete Schlüssel und Bobs digitale Signatur des verschlüsselten Textes.

Das RSA-Verfahren: Dieses nach seinen Erfindern Rivest, Shamir und Adleman bezeichnete Verfahren der public-key-Kryptographie wurde erstmals im Jahre 1978 veröffentlicht und steht seit über 20 Jahren im Praxiseinsatz.

3 ERZEUGUNG EINES RSA-SCHLÜSSELPAARES

Das Schlüsselpaar, welches aus *public key* und *private key* besteht, wird durch mathematische Operationen erzeugt. In diesem Text werden ausser den für die Grundrechenarten verwendeten Rechenzeichen + - * / noch folgende Rechenzeichen verwendet:

^	für die Potenzierung
%	für die modulo-Operation (Rest der ganzzahligen Division)

Man nehme zwei große Primzahlen `p` und `q` und berechne die Zahl `n` als ihr Produkt:

[5] Dieser Text wurde von Martin Karl Unger verfasst.

```
n = p * q
```

Man berechne den Wert der Euler'schen phi-Funktion `phi(n)` als

```
phi(n) = (p-1) * (q-1)
```

Man nehme eine beliebige Zahl, die teilerfremd ist zu `phi(n)`, zum Beispiel eine Primzahl, und bezeichne sie als `e` und berechne dann eine Zahl `d` sodass gilt:

```
(e * d) % (phi(n)) = 1
```

Die Zahl `d` bezeichnet man als *private key*. (Merkhilfen: domestic key, decrypt). Die Zahl `e` bezeichnet man als *public exponent*. (Merkhilfen: external key, encrypt). Die Zahl n bezeichnet man als *modulus*. Das Zahlenpaar `(e, n)` wird als *public key* veröffentlicht.

Jetzt gelten für jede ganze Zahl t die Gleichungen:

((t ^ e) %n) ^ d) %n = t
((t ^ d) %n) ^ e) %n = t

oder unter Weglassung der überflüssigen Klammern:

```
(t^e %n)^d %n = t
(t^d %n)^e %n = t
```

Der mithilfe der sogenannten Restklassenarithmetik zu führende Beweis für die Gültigkeit der beiden obigen Gleichungen kann in [1, 2] nachgelesen werden.

4 VER- UND ENTSCHLÜSSELUNG MIT RSA

Der zu verschlüsselnde Ausgangstext wird in eine Folge von Zahlen verwandelt. Jede Zahl `a` aus dieser Folge von Zahlen wird mit dem *public exponent* `e` des Adressaten potenziert und mit dem *modulus* `n` des Adressaten reduziert:

```
c = a^e %n
```

Diese Zahlen `c` werden in Zeichen verwandelt, die an den Adressaten versendet werden. Der Empfänger verwandelt diese Folge von Zeichen in eine Folge von Zahlen `c` und berechnet

```
a = c^d %n
```

und kann dann diese Folge von Zahlen `a` wieder in eine Folge von Zeichen, also den Ausgangstext, umwandeln.

5 DIGITALE SIGNATUR MITHILFE DES RSA-VERFAHRENS

Der Unterzeichner berechnet mithilfe einer kryptographischen Hash-Funktion wie SHA1 oder MD5 den Hashwert h des zu unterschreibenden Textes. Begriffserklärungen:

Eine Hash-Funktion bildet einen großen Wertebereich (z. B. die Menge aller möglichen Texte) auf einen weniger großen Wertebereich (z.B. die Menge aller Bit-Folgen mit 128 oder 160 Bit Länge) ab.

Eine kryptographische Hash-Funktion ist eine in der Kryptographie verwendete Einweg-Funktion.

Eine Einweg-Hash-Funktion ist so beschaffen, dass aus einem Hash-Wert keinen zu diesem Hash-Wert passenden Text konstruieren kann, ohne alle möglichen Texte auszuprobieren. Dadurch ist es fast unmöglich, den unterschriebenen Text derart zu ändern, dass er nach der Änderung noch zur digitalen Signatur passt.

Dann verschlüsselt der Unterzeichner die Zahl `h` mit seinem *private key*, um die digitale Signatur `s` zu erhalten:

```
S = H^D %N
```

Versendet werden Ausgangstext, digitale Signatur `s`, Bezeichnung der Hashfunktion und ein Zertifikat, welches den *public key* (das Zahlenpaar `(e,n)`) des Un-

terzeichners enthält. Zur Überprüfung dieser digitalen Signatur kann jeder Empfänger selber den Hashwert h berechnen und prüfen, ob die Gleichung

```
h = s^e %n
```

erfüllt ist.

6 BLINDE SIGNATUR MIT HILFE DES RSA-VERFAHRENS

Das Applet erhält vom Server das Zertifikat desjenigen Wahlsprengels, in welchem der Wähler wahlberechtigt ist. Dieses Zertifikat enthält den *public key* `(e, n)` des Wahlsprengels. Das Applet nimmt eine Zufallszahl `t` und eine zu `n` teilerfremde Zahl `r` und berechnet `x` als

```
x = (t * (r^e %n)) %n
```

und schickt diese Zahl `x` an den *Registration Server*, also an den Rechner der Wahlbehörde. Der *Registration Server* berechnet die Zahl `y` als

```
y = x^d %n
```

und retourniert diese Zahl `y` an das Applet. Das Applet ermittelt eine Zahl `r1` sodass gilt:

```
(r * r1) %n = 1
```

Die Regeln der Restklassenarithmetik erlauben folgende Umformungen:

```
y = x^d %n
```

und wegen

```
x = (t * (r^e %n)) %n
```

```
y = (t * r^e)^d %n
```

```
y = (t^d * ((r^e)^d %n)) %n
```

```
y = (t^d * r) %n
```

```
(y * r1) %n = (t^d * ((r * r1) %n)) %n
```

```
(y * r1) %n = t^d %n
```

Daher multipliziert das Applet die Zahl `y` mit der Zahl `r1` und reduziert das Ergebnis modulo `n`, um den Wert des Ausdrucks `(t^d %n)` zu erhalten:

```
t^d %n = (y * r1) %n
```

Der soeben indirekt berechnete Wert des Ausdrucks `(t^d %n)` wird als `td` bezeichnet und bildet gemeinsam mit der Zahl `t` einen Teil der elektronischen Wahlkarte.

Fazit: Bei der Wahl kann die elektronische Wahlurne prüfen, ob die Gleichung

```
t = td^e %n
```

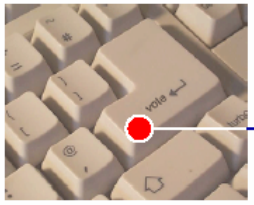
erfüllt ist, aber sie kann aus den Zahlen `t` und `td` nicht auf die Identität des Wählers schließen. Damit bleibt das Wahlgeheimnis gewahrt, obwohl sich bei der Ausgabe der elektronischen Wahlkarte jeder Wähler gegenüber dem Rechner der Wahlbehörde identifiziert hat.

7 LITERATUR

[1] Buchmann, Johannes: Einführung in die Kryptographie, Springer 1999

[2] van Tilborg, Henk: Fundamentals of Cryptology: a professional reference and interactive tutorial, Kluwer Academic Publishers 2000

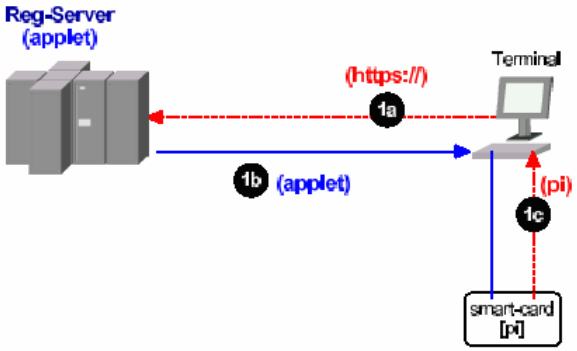
2.4 Prozess



www.e-Voting.at

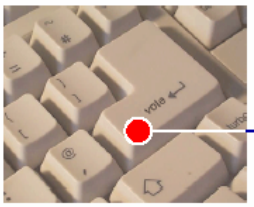
Alexander Prosser / Robert Krimmer / Robert Kofler / Martin Unger

1



smart-card

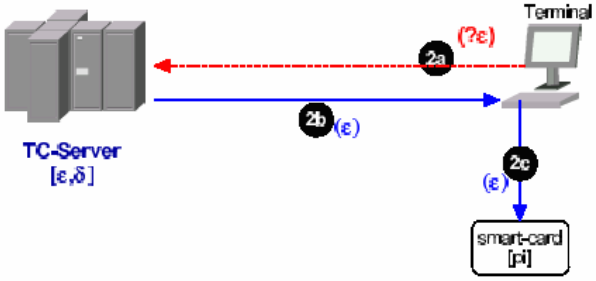
pi



www.e-Voting.at

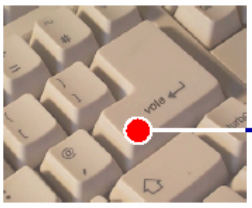
Alexander Prosser / Robert Krimmer / Robert Kofler / Martin Unger

2



smart-card

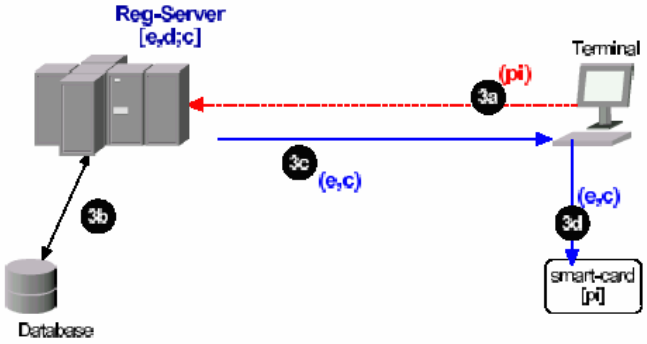
pi,
ε



www.e-Voting.at

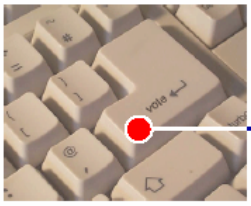
Alexander Prosser / Robert Krimmer / Robert Kofler / Martin Unger

3



smart-card

pi,
ε,
e,
c

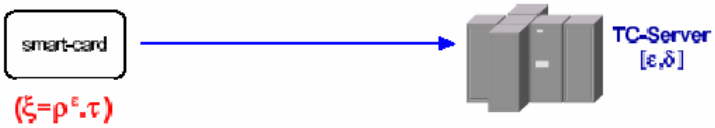


www.e-Voting.at

Alexander Prosser / Robert Krimmer / Robert Kofler / Martin Unger

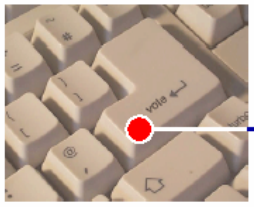
4a

Smart-Card generiert Zufallszahlen ρ,τ.
Verschlüsselt ρ mit ε , blindet (multipliziert) mit τ und erhält $\xi=\rho^{\varepsilon}.\tau$



smart-card

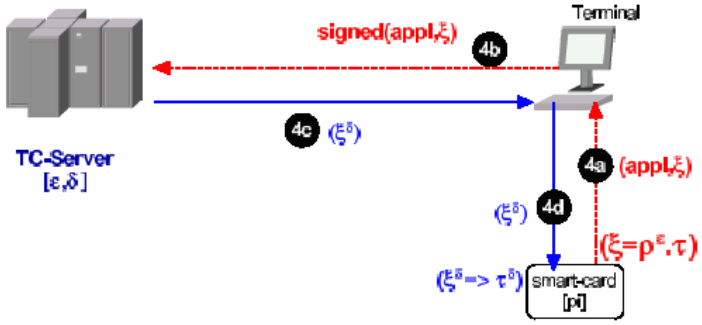
pi,
τ, ρ, ε,
e,
c



www.e-Voting.at

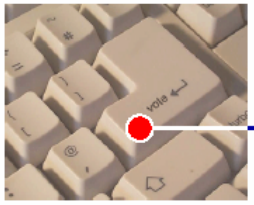
Alexander Prosser / Robert Krimmer / Robert Kofler / Martin Unger

4b



smart-card

pi,
τ, ρ, ε,
e,
c

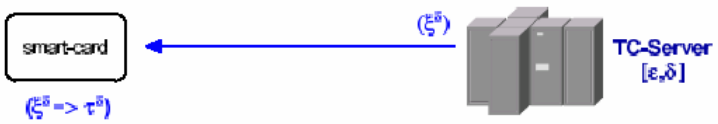


www.e-Voting.at

Alexander Prosser / Robert Krimmer / Robert Kofler / Martin Unger

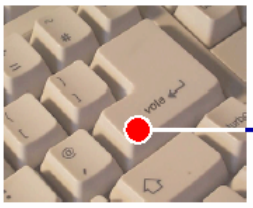
4c

Smart-Card dividiert ε^{δ} durch ρ und erhält τ^{δ}



smart-card

pi,
τ, ρ, ε,τ^δ
e,
c

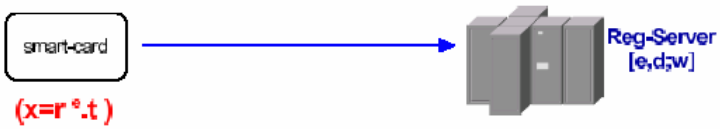


www.e-Voting.at

Alexander Prosser / Robert Krimmer / Robert Kofler / Martin Unger

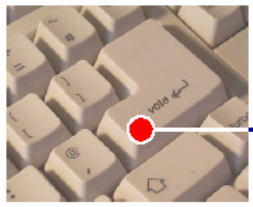
5a

Smart-Card generiert Zufallszahlen r ,t .
Verschlüsselt r mit e , blindet (multipliziert)
mit t und erhält x=r °.t



smart-card

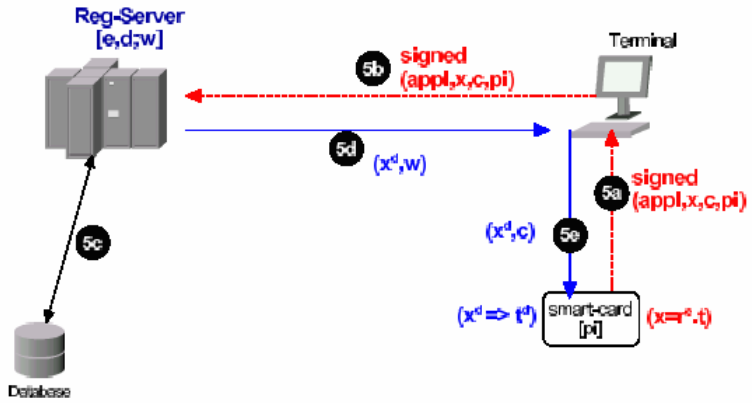
pi,
τ, ρ, ε,τ$^{\delta}$
r,t,e,
c



www.e-Voting.at

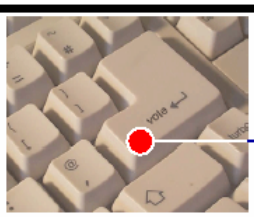
Alexander Prosser / Robert Krimmer / Robert Kofler / Martin Unger

5b



smart-card

pi,
τ, ρ, ε,τ$^{\delta}$
r,t,e,
c



www.e-Voting.at

Alexander Prosser / Robert Krimmer / Robert Kofler / Martin Unger

5c

Smart-Card dividiert x^d durch r und erhält t^d

smart-card

(x^d)

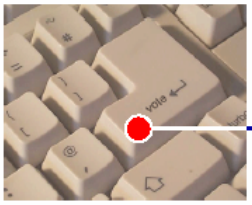
Reg-Server
[e,d;w]

smart-card

$(x^d => t^d)$

smart-card

pi,
$\tau, \rho, \varepsilon, \tau^\delta$
r,t,e,t^d,
c



www.e-Voting.at

Alexander Prosser / Robert Krimmer / Robert Kofler / Martin Unger

6

Smart-Card enhält nun
Wahlberechtigung (t, t^d) und
Prüfkarte (τ, τ^δ)

smart-card

(x^d)

Reg-Server
[e,d;w]

smart-card

$(x^d => t^d)$

smart-card

pi,
$\tau, \rho, \varepsilon, \tau^\delta$
r,t,e,t^d,
c

3 Publikationen

3.1 Vom e-Government zur elektronischen Demokratie

Abstract: [6] *Elektronische Transaktionen über das Internet, insbesondere das World Wide Web sind zu einem entscheidenden Faktor im täglichen Wirtschaftsleben geworden. In der nahen Vergangenheit begann auch der öffentliche Sektor das Internet für die Abwicklung von administrativen Transaktionen zu entdecken. Diese Arbeit analysiert, wie ein elektronisches Wahlsystem unter Einhaltung der allgemeingültigen Wahlgrundsätze in Österreich realisiert werden kann und im Besonderen wird dabei auf Probleme in Zusammenhang mit der Verwendung einer Signaturkarte eingegangen.*

1 VORAUSSETZUNGEN FÜR E-VOTING

Für die Durchführung von Wahlen über das Internet gelten auch die allgemeinen Wahlrechtsgrundsätze entsprechend Art 26 B-VG, die von einem gleichen, unmittelbaren, geheimen und persönlichen Wahlrecht[7] ausgehen. Als spezielle Vorraussetzungen für elektronische Wahlen kann man die Ausarbeitung des Internet Policy Institutes heranziehen [IPI2001]:

- **Freie Wahlen:** jeder Wähler muss in der Lage sein sein/ihr Wahlrecht geheim und ohne Einflussnahme durch einen Dritten abgeben zu können. Die Stimme muss unverändert in der Wahlurne ankommen.

- **Geheime Wahl**: Keine Person darf Kenntnis von der abgegebenen Stimme einer anderen Person erlangen können; die Auszählung der Stimmen muss solange verzögert werden, bis eine genügende Anzahl an Stimmen abgegeben wurde, sodass kein Rückschluss auf das Wahlverhalten eines Einzelnen möglich ist (dies ist vergleichbar mit einer Mindestgröße der Sprengel).

- **Gleiche Wahl**: Jede Stimme muss dasselbe Gewicht haben – Keine Stimme darf aufgrund technischer Schwierigkeiten verloren gehen oder ungültig werden. Des Weiteren darf das Recht zu Wählen nicht weiter eingeschränkt werden als in den gesetzlichen Vorgaben.

- **Nachvollziehbarkeit**: Der gesamte Wahlprozess muss nachvollziehbar und wiederholbar sein.

2 GRUNDPROBLEM: IDENTIFIKATION UND ANONYMITÄT

Während die Wahlgrundsätze der Gleichheit und Unmittelbarkeit durch organisatorische Maßnahmen leicht zu bewältigen sind, leitet sich aus den Grundsätzen des persönlichen und geheimen Wahlrechts das technische Grundproblem ab: Wie kann ich im Rahmen eines elektronischen Transaktionsprozesses mich eindeutig identifizieren und dann eine anonyme Stimme abgeben?

Von technischer, wie auch rechtlicher Sicht lehnt man sich bei den Spezifikationen für eine elektronische Wahl an der bewährten (aber in Europa nur in Deutschland, Frankreich und der Schweiz erlaubten) Briefwahl an. [DUJ00]. In Analogie identifizierten *Nurmi*, *Salomaa* und *Santean* bereits 1991 [NSS91] im Rahmen einer eingehenden Analyse des Wahlverfahrens zwei unterschiedliche Vorgänge:

- Der Registrierungsschritt, während dem sich der Wähler eindeutig identifiziert

- Die Stimmabgabe, die anonym erfolgt

Obwohl e-Voting ein verhältnismäßig neues Konzept ist, wurden in den letzten Jahren zahlreiche Systeme entwickelt, die bereits am Markt angeboten werden

[6] Dieser Artikel von Prosser, Kofler und Krimmer erschien im Sammelband „IT in Recht und Staat“, herausgegeben von Schweighofer, Menzel und Kreuzbauer, Verlag Österreich 2002

[7] Für eine weitergehende Behandlung der rechtlichen Rahmenbedingungen in Österreich für e-Voting siehe den Beitrag von Menzel in dem gleichen Sammelband.

bzw. sich noch im Prototypenstadium befinden. Die Systeme verfolgen dabei die unterschiedlichsten Lösungsansätze, doch haben sich drei Methoden zur Identifikation des Wählers herauskristallisiert:

- PIN-basierte Systeme
  Der Wählende arbeitet hier als identifizierter Benutzer über das Internet, nach dem Login kann der Wahlzettel ausgefüllt und abgeschickt werden, wobei die Kommunikation zwischen Browser und Wahlserver kryptographisch gesichert ist.[8]

- TAN-basierte Systeme
  Bei solchen Systemen werden TANs ausgegeben, die Wahl ist üblicherweise über eine Internetschnittstelle unter Angabe des TAN möglich. Der Webverkehr zwischen Wählendem und Wahlserver erfolgt verschlüsselt, der Schlüssel wird von einem Trust Center [FFW99], [FiWh00] zugebracht. Der Wählende erhält dabei eine Zufallszahl als Quittung, mit der man an einem anderen Terminal überprüfen kann, ob die Stimme korrekt gezählt wurde.

  Bei einer Kollusion der beiden Stellen zur TAN-Ausgabe und Stimmabgabe, kann natürlich die Anonymität nicht mehr gewährleistet werden. Als weiterer Nachteil kommt die physische Distribution von TANs pro Wahlakt.

  Aufgrund der bisherigen Diskussion stellen die Autoren die falsifizierbare Hypothese auf, dass weder PIN- noch TAN-basierte Systeme für den Einsatz bei Wahlen zu Vertretungskörpern von Körperschaften öffentlichen Rechts über das Internet geeignet sind, da die Sicherung der Anonymität weder bei TANs noch bei PINs technisch garantiert werden kann, sondern von der Integrität der Wahlorganisatoren bzw. ihrer technischen Erfüllungsgehilfen abhängt; sowie PIN und TAN als Identifikationsmittel keine gesetzliche Grundlage aufweisen

- Die weitere Diskussion konzentriert sich daher auf die dritte Variante, die Realisierung mittels signaturkartenbasierte Systeme unter Verwendung kryptograhischer Verfahren.

3 WAHLALGORITHMEN

3.1 Verfahren nach [FOO93]

Für die Sicherstellung der Anonymität haben *Fujioka*, *Okamoto* und *Ohta* in ihrem Ansatz die Wahlbehörde in zwei voneinander unabhängige Einheiten (Server) unterteilt:

- Registrierungsserver

- Urnenserver

Das einphasige Verfahren nach [FOO93] sieht vor, dass der gesamte Prozess in einem Vorgang abgewickelt wird. Es beginnt damit, dass der Wähler seinen Stimmzettel ausfüllt und dann mit einem symmetrischen Schlüssel verschlüsselt. Anschließend bereit er den verschlüsselten Stimmzettel auf die blinde Signatur vor[9]. Diesen unterschreibt er und schickt es an den Registrierungsserver **<k (SZ)>**. Dort überprüft der Server die Identität des Wählers anhand der digitalen Signatur und wenn sie gültig ist, unterschreibt er den Stimmzettel. Darauf wird das blind signierte Dokument dem Wähler zurückgeschickt **<b sig (k (SZ))>** und vom ihm entschlüsselt. Nun besitzt der Wähler einen verschlüsselten, vom Registrierungsserver unterschriebenen, Stimmzettel, den er an den Urnenserver schickt

[8] Das hier diskutierte Verfahren der PIN-basierten Identifikationssysteme ist nicht zu verwechseln mit der Eingabe von PIN für die Zugriffssicherung bei Signaturkarten

[9] Das Verfahren der blinden Signatur wird im allgemeinen Sprachgebrauch mit der Leistung der Unterschrift auf einem Blaupapierkuverts verglichen. Für eine eingehendere Erklärung siehe [Chau82]

<b sig (k (SZ))>. Dieser überprüft die Unterschrift auf dem Stimmzettel, ob diese auch wirklich vom Registrierungsserver stammt. Nach erfolgter positiver Prüfung schreibt der Urnenserver beim ursprünglichen Protokoll die verschlüsselten Stimmzettel auf eine Liste, die nach Ende der Wahl veröffentlicht wird. Der Wähler kann dann überprüfen ob sein verschlüsselter Stimmzettel auf der Liste ist, und wenn ja, dann schickt er dem Urnenserver den Entschlüsselungsschlüssel **<k'>**. Diesen verwendet der Server anschließend um die Stimmzettel zu entschlüsseln und auszuzählen. Nach Ende der Auszählung werden die Schlüssel und die Stimmzettel wiederum auf einer öffentlichen Liste publiziert, damit jeder Wähler unabhängig das Wahlergebnis überprüfen kann.

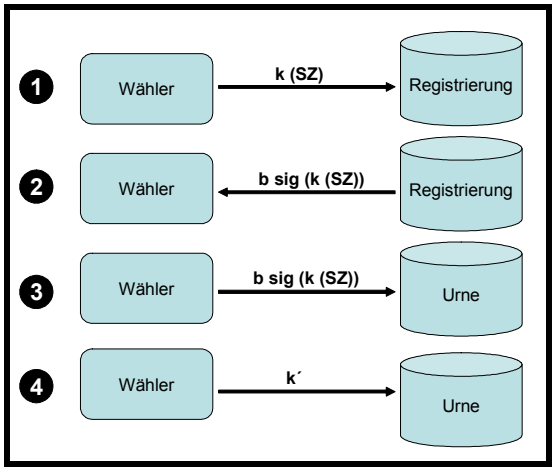


Abbildung 1: Ablaufdiagramm [FOO93]

Dieses Verfahren wurde bereits vielfach variiert und hat doch in allen Varianten ein grundlegendes Problem: Es ist als Ein-Phasen-Algorithmus konzipiert, das bedeutet, dass die zwei Schritte Identifizierung und Wahlvorgang in einem erledigt werden. Das ermöglicht, wenn der Registrierungs- und der Urnenserver zusammenarbeiten, das Brechen der Anonymität als auch das Einschleusen von Stimmen für wahlberechtigte Personen, die nicht zur Wahl gegangen sind.

3.2 Verfahren nach Prosser/Müller [PrMü01]

Auch dieser Algorithmus geht von den zwei unabhängigen Wahlbehördeeinheiten Registrierungs- und Urnenserver aus. Allerdings gibt es einen grundsätzlichen Unterschied, denn hier wird basierend auf der Analyse von [NSS91] der Wahlprozess in zwei nacheinander ablaufenden, aber zeitlich unabhängigen, Phasen getrennt:

- Phase (1): Die Registrierungsphase, während der der Wähler identifiziert ist, wobei sich der Wähler zusätzlich bei seinem Trust Center eine blind signierte Bestätigung holt.
- Phase (2): Die Stimmabgabenphase, bei der die anonyme Stimme abgegeben wird und bei der der Wähler nicht identifiziert werden kann

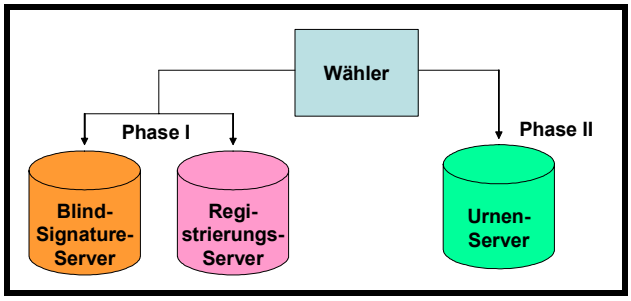


Abbildung 2: Die beteiligten Parteien

3.2.1 Die Registrierungsphase

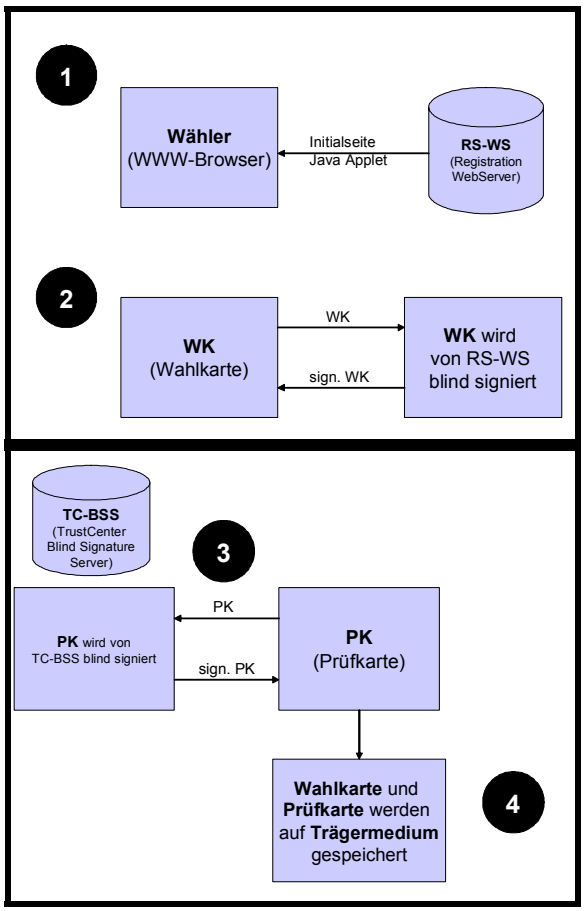


Abbildung 3: Ablaufdiagramm Registrierungsphase

Zuerst generiert der Wähler ein Token und bereitet sie auf die blinde Unterschrift vor, das er dann mit der digitalen Unterschrift seiner Signaturkarte unterschreibt. Dieses Paket schickt er nun an den Registrierungsserver, der seine Wahlberechtigung aufgrund der Unterschrift überprüft, dieses blind unterschreibt und an den Wähler retourniert. Der Wähler entschlüsselt das Paket und hat damit ein vom Server unterschriebenenes Token bzw. eine von der Wahlbehörde unterschriebene elektronische Briefwahlkarte.

Anschließend wird der selbe Vorgang wie mit dem Registrierungsserver mit dem Blind Signature Server im Trust Center wiederholt um sich die elektronische Briefwahlkarte auch vom ihm blind unterschreiben zu lassen. Dies hat den Zweck, dass die Wahlbehörde keine gefälschten Stimmen abgeben kann, selbst wenn der Registrierungsserver und der Urnenserver zusammenarbeiten, da sie ja immer auch die Unterschrift vom unabhängigen Blind Signature Server benötigen.

Am Ende der Registrierungsphase hat der Wähler eine elektronische Briefwahlkarte, die sowohl vom Registrierungsserver als auch vom Blind Signature Server im Trust Center blind unterschrieben wurde. Nachdem die Phase 1 zeitlich unabhängig von der Stimmabgabephase ist, muss diese Wahlberechtigung auf einem Trägermedium zwischengespeichert werden und auf die Anonymität des Mediums wert gelegt werden.

3.2.2 Stimmabgabephase

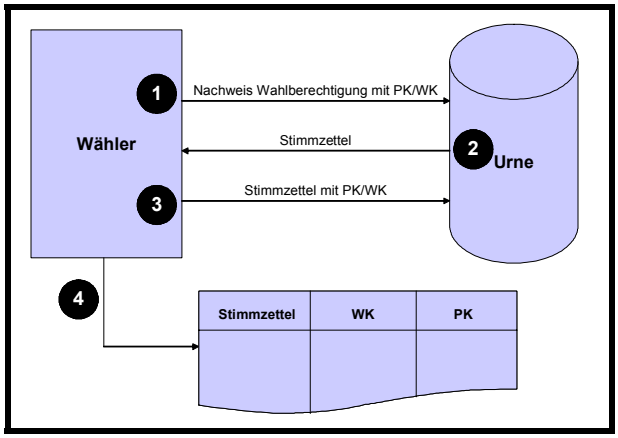


Abbildung 4: Ablaufdiagramm Stimmabgabephase

Die zweite Phase des Wahlprozesses findet am eigentlichen Wahltag statt. Damit der Wähler wählen kann, verwendet er seine elektronische Briefwahlkarte (Token), die auf dem Trägermedium gespeichert ist. Er schickt diese an den Urnenserver, der die Validität mittels der öffentlichen Schlüssel von Registrierungs- und Blind Signature Server überprüft. Sind beide gültig,

schickt er dem Wähler einen Stimmzettel. Das wird dann vom Wähler ausgefüllt und zusammen mit der Wahlberechtigung an den Urnenserver geschickt. Nach Ende der Wahl veröffentlicht der Urnenserver eine Liste mit Stimmzettel und zugehöriger elektronischer Briefwahlkarte.

3.2.3 Die Zwischenspeicherung und das Trägermedium

Die Wahl des Trägermediums ist der kritische Faktor für die Durchführbarkeit einer Wahl basierend auf dem Prosser/Müller Algorithmus. Für diese Zwischenspeicherung der elektronischen Briefwahlkarte (Token) sehen wir grundsätzlich drei Möglichkeiten:

(a) auf der Signaturkarte

Vorteil der Speicherung des Token auf der Signaturkarte des Wahlberechtigten ist der Schutz vor Datenverlust gegenüber konventionellen Speichermedien und der Schutz vor dem Auslesen, wenn das Token durch einen PIN gesichert wird. Diese Variante hat aber zwei wesentliche Schwierigkeiten:

- zum einen muss das Signaturgesetz des betreffenden Landes einen frei beschreibbaren Bereich in einer Signaturkarte überhaupt zulassen (in der EU ist dies durch die Signaturrichtlinie gegeben) und der betreffende Service Provider muss diesen Bereich für die Benutzung tatsächlich vorsehen bzw. die Karte dafür auch zertifiziert sein.

- zur Sicherung der Anonymität beim Senden des Tokens an die Urne, darf es keine frei auslesbaren, nicht PIN-gesicherten Informationen auf der Karte geben, die Rückschlüsse auf den Karteninhaber
  zulassen, die ohne Zugriffssicherung ausgelesen werden können.

Der letzte Punkt ist dabei eine entscheidende Einschränkung und stellt sich vollkommen unabhängig, ob Registrierung und Stimmabgabe in einer oder zwei Phasen erfolgen. Regelmäßig wird nämlich von den Providern der Name des Karteninhabers ungeschützt im Klartext auf der Karte gespeichert, außerdem hat jede Signaturkarte der Welt eine eindeutige Seriennummer, die bei Erzeugung der Karte gesetzt wird, wobei diese Seriennummer ein Tracking der Karte von der Initialisierung beim Trust Center-Betreiber über die Ausgabe an den Kunden bis hin zum Karteninhaber zulässt. Selbst wenn die Software des Wahlbetreibers im Sourcecode offen gelegt wird und dieser Code nachweislich nicht auf die Kartenseriennummer oder andere Klartextinformation auf der Karte zugreift, wird dies als Absicherung in einem solch sensiblen Bereich vermutlich kaum reichen, um Akzeptanz beim Bürger zu erreichen. Alternativ wäre auch eine Modifikation des Betriebssystems der Signaturkarte denkbar, wodurch ein Auslesen dieser Seriennummer verhindert werden könnte bzw. mit einem zusätzlichen Zugriffsschutz (PIN) versehen würde.

(b) auf einem beliebigen Datenträger analog zu einer elektronischen Geldbörse

Diese Variante löst sofort die eben diskutierten Probleme mit Seriennummer und Klartextinformation: bei Registrierung legt der Wählende beispielsweise eine Diskette oder einen USB-Schlüssel ein und das Token wird darauf zwischengespeichert. Die Implementierung wäre einfach und benötigte ausschließlich allgemein verfügbare Infrastruktur, wie Diskettenlaufwerke oder USB-Ports. Der Nachteil dieser Lösung besteht im unsicheren und gegen Kopieren und Auslesen keinerlei geschützten Medium. Sichere USB-Schlüssel sind derzeit in Entwicklung aber noch nicht ausgereift.

(c) auf einer anderen Prozessor-Chipkarte als der Signaturkarte

Die Signaturkarte wird ausschließlich zur Identifikation beim Registrator eingesetzt, die Speicherung der elektronischen Briefwahlkarte erfolgt auf der zweiten Karte und wird durch End-to-End-Verschlüsselung gesichert. Bei der Stimmabgabe im Dialog mit der Urne wird ausschließlich die zweite Karte eingesetzt.

Die Nachteile dieser Variante bestehen im komplizierteren Handling (während des Registrierens muss die Karte umgesteckt werden) und in der Tatsache, die Karte mit der Wahlberechtigung nun nicht mehr einer Person zugeordnet ist, wie dies bei der Signaturkarte der Fall war.

4 SCHLUSSWORT

Dieser Beitrag zeigt, dass Design und Realisierung eines e-Voting-Systems komplex und keinesfalls trivial sind, wenn die Einhaltung der Wahlrechtsgrundsätze bei der Abwicklung der Wahl technisch garantiert werden sollen. Erschwert wird die Implementierung durch die offensichtliche Nichteignung der vorhandenen Signaturkarten für die Zwischenspeicherung bzw. den Einsatz einer anonymen Wahlberechtigung, die erst durch ein entsprechendes Redesign überwunden werden kann; insbesondere indem dem Anwender die volle Kontrolle darüber gegeben wird, welche Daten von einer Applikation ausgelesen werden, die auf die Karte zugreift.

5 LITERATUR

[Chau82] *Chaum, D.*: Blind Signatures for Untraceable Payments in: Chaum, D., Rivest, R.L., Sherman A.T. (Hrsg): Advances in Cryptology, Proceedings of Crypto 82, S. 199-203

[DUJ00] *Dujmovits, W*: Auslandsösterreicherwahlrecht und Briefwahl, Verlag Österreich, Vienna 2000

[FFW99] *Feghhi, J., Feghhi, J., Williams, P.*: Digital Certificates – Applied Internet Security; Addison-Wesley, Reading, 1999

[FiWh00] *Fish, E.A., White, G.B.*: Secure Computers and Networks, Analysis, Design and Implementation; CRC Press, Boca Raton, 2000

[FOO93] *Fujioka, A., Okamoto, T., Ohta, K.*: A Practical Secret Voting Scheme for Large Scale Elections; Advances in Cryptology – AUSCRYPT92, Springer-Verlag, Berlin, 1993, S. 244-251

[IPI01] Internet Policy Institute; Report on the National Workshop on Internet Voting downloadable unter http://www.internetpolicy.org/research/e_voting_report.pdf (13.05.2001)

[NSS91] *Nurmi, H., Salomaa, A., Santean, L.*: Secret ballot elections in computer networks; Computers and Security 36 (10), 1991, S. 553-560

[ProMü01] *Prosser, A., Müller, R.*: Vom E-Government zur E-Democracy, eingereicht bei Wirtschaftsinformatik

3.2 Electronic Voting: Algorithmic and Implementation Issues

Abstract:[10] *Electronic Transactions over the Internet, particularly using the World Wide Web have become an integral part of economic life. Recently also the public sector has started to use the new medium for its administrative processes. This paper analyses several approaches to implement an electronic voting system and discusses them with a view to voter anonymity and protection from manipulations.*
The paper then develops an algorithm designed to guarantee anonymity of the voter and to avoid the risk of manipulation of votes. The algorithm is based upon the strict separation of voter registration and submission of votes, which implies that certain information has to be stored on a secure media. The paper discusses the security criteria and possible implementation options for such secure storage.

1 INTRODUCTION

In the past years many governments have started to adopt Internet-based applications for their administrative processes; applications range from the simple download of forms to Internet-based submission of applications or tax declarations to full-scale electronic procurement systems. The first mover in this area was the U.S. Federal Government with the Federal Acquisition Network (for an analysis see [1]), in the meantime similar systems are being deployed in several EU countries [2]. [3].

Such systems use commercially available technology and basically automate administrative processes. The question arises whether Internet services could also be used for voting processes and, if so, to derive the design principles of such systems.

Let us first clarify some notions:

- - We consider electronic government to be the application of Internet technology to support administrative processes; Internet technology comprises Web applications, electronic mail, electronic data interchange as well as mobile access to the Internet via WAP or UMTS terminals.
- Electronic democracy would be the use of such technology to support the participation of the citizen in democratic decision-making, such as elections, petitions, referendums, and the advertising prior to them,
- of which electronic voting (e-voting) is a subset, that is the support of an anonymous voting procedure, which is the main theme of this paper.

The term “electronic voting” has been used for a large variety of systems, ranging from hand-held infrared devices used, for instance, at the shareholder meeting of a publicly listed corporation, kiosk systems with touch screens to be used in polling stations to remote voting via the Internet. In the following, we will focus on Internet voting only. It should be noted, that any Internet voting system can also be used as a kiosk system in the polling station.

The following Section 2 analyzes the requirements for an electronic voting system (2.1 and 2.2) and discusses several approaches to implement electronic voting found in the literature (2.3 and 2.4); Section 3 proposes an alternative algorithm, which is the basis for an e-voting system currently developed for the City of Vienna (Austria). Implementation issues resulting from the requirements of the algorithm are also discussed in Section 3.3. Section 4 outlines further research.

[10] This article was written by Kofler, Krimmer and Prosser and accepted for the Hawaii International Conference on Information Systems 36 from January 6th – 9th, 2003

2 REQUIREMENTS FOR E-VOTING

2.1 Voting Principles

In General, the requirements for conventional, "paper-based" voting also apply to e-voting. These principals for democratic elections can be expected to be universal; of course, voting procedures may differ in many details. Taking the Austrian Federal Constitution as an example (Art. 26 B-VG) the right to vote is defined as common, direct, equal, personal and secret. The Internet Policy Institute [4] specified these principles with a view to electronic voting:

- Free elections: the citizen must be able to use her/his voting rights without being coerced and without undue influence of a third party. The vote must reach the election authority without the chance of manipulation.

- Secret voting: no person must know the vote of another person, counting of votes must be delayed until a sufficient number of votes ensures that no conclusions as to the vote of the individual voter can be drawn.

- Equal voting rights: each vote must have the same weight. No vote must become invalid by predictable technical problems or must be lost on its way to the voting authority. Also, the right to vote must not be made dependent on factors other than those enumerated in the Law (e.g., a criminal conviction).

- Audibility: the whole voting process must be transparent and reproducible.

- Reliability: the whole voting system should work robustly (i.e. so that no votes are lost), even if failures occur like loss of Internet communication or malfunctioning voting machines.

- Flexibility: the system should be configurable for many different election scenarios (like different ballot question formats or multiple languages etc.) and on a technical level compatible with multiple operation system platforms as well.

- Uniqueness: No voter should be able to vote more than once.

- Integrity: votes as such should not be modifiable, forged or deleted without detection and the possibility to repair the manipulation.

- Convenience: election systems should not require extra skills to be usable and without unreasonable need for equipment.

2.2 Authentication vs. Anonymity

As it is rather uncomplicated to fulfill the requirements for equal and direct elections, the main technical problem for the deployment of e-voting can be derived from the requirement for personal and secret elections. How can a person be identified unequivocally and her credentials checked in the voting process and yet be able to cast an anonymous vote? Any e-voting protocol has to solve this central issue.

Internet-based e-voting can best be compared to mail voting, which has already existed for quite some time in some European countries, for instance, in Germany and France; recently it was also introduced in Switzerland; Austria currently limits the use of mail voting to citizens living abroad [5]. In mail voting, the voter has to register before the elections and can cast her vote within a given timeframe anywhere. The vote is sent to the election authorities by mail. In many respects Internet voting can be seen as an analogy.

In a fundamental contribution, *Nurmi*, *Salomaa* and *Santean* [6] identified the two basic elements in any e-voting system:

1) The registration process during which the voter is identified unambiguously

2) The voting process as such, where the voters is anonymous

2.3 Identification

The systems can be grouped in 3 different classes: PIN-based or TAN-based systems and systems using smart cards for identification.[11]

2.3.1. PIN-based Systems:

The voter is an identified user on the Internet, after Login the ballot sheet can be filled out and sent in, where the communication between the browser and the voting server is secured using cryptographic standards, however, it is obvious that anonymity cannot be guaranteed.

As an example, the system of Soundcode called "Vote here gold" [7] offers a PIN-based system that can be used via the Internet. The voter works as an identified user that can fill out and hand in the ballot sheet after a login, where the communication between WWW browser and election server is secured cryptographically. Using this system, anonymity can obviously not be guaranteed to the voter and it has to be relied on the integrity of the election authority. The main applications for such a system are (i.e. student) union, faculty or interest group elections where absolute anonymity is not necessary, the most important issue or where it is planned to hold open and public elections. Such systems can lower the transaction costs for elections drastically and in the case of dislocated voters be prerequisite for a fast election. An example could be a strike ballot of an airline.

2.3.2 TAN-based Systems

In such systems Transaction Numbers are issued and the election is usually possible by using the TAN in a Web browser. The connection between the voter and the Web server is also secured and the cryptographic key is issued by a Trust Center (for an introduction, see [8], [9]). The voter receives a random number as a receipt for casting the vote, which can be used to check whether the vote entered the tally correctly at a different Website.

If the party issuing the TANs and the party running the election server collecting the votes collude, anonymity cannot be guaranteed. Also, the distribution of TANs for each single election is costly and errors may occur.

The election.com system is an example for a TAN-based system that was used in May 2002 to elect the EU-Student-council [10]. In this case TANs are used and the election itself is possible by entering the TAN using an Internet-application. The Web-traffic between the voter and the election server is securely encrypted and the key is provided by a Trust Center.

The voter receives a random number as a receipt, with that it is possible to check at a different terminal if the vote was counted correctly.

A similar system was used for the election of the Jugendgemeinderat (young city council, an unofficial advisory board) at the German town of *Fellbach* in 2001 [11].

2.3.3 Smart Card-based Systems

Hence, neither PIN- nor TAN based systems can be used for democratic elections, however, both are relatively easy to implement and can be used for a wide

[11] The PIN-based identification method is to be differentiated from the entry of a PIN for securing access to data stored on a smart card, which is discussed later.

range of voting applications, where requirements for anonymity are less stringent (e.g., workers representatives, chambers or professional organizations) or where anonymity is not a requirement at all (e.g., in the Swiss canton Appenzell Innerrhoden votes are traditionally still cast openly at the commencement of the *Landsgemeinde*, where all persons entitled to vote come together to elect the *Ständerat* [12]).

Anonymity can not be guaranteed technically for any of the two systems, but is dependent on the integrity of the election organizers and their server administration. Also, neither PIN nor TAN have a legal foundation as means of identification as is the case with digital signatures. Hence, the further discussion will concentrate on e-voting systems using smart cards for digital signatures, which also enables the use of cryptographic methods.

2.4 One-stage Smart Card-based Systems

The algorithm by *Fujioka*, *Okamoto* and *Ohta*, which was first published in 1993 [13], has become the algorithmic basis of a considerable number of systems (for example Lorrie Cranor's Sensus [14] or the German system i-Vote [15]).

Let us first introduce some notation:

BS Ballot Sheet

B Ballot box Server

R Registration Server

V voter

m , m' Symmetric crypto key

$S^{\{V,R,B\}}_{\{priv,pub\}}$ The voter's, the registration's and the ballot box server's signature key pair

$K^{\{V,R,B\}}_{\{priv,pub\}}$ Their key pair for encryption

The algorithm assumes the use of a Trust Center for obtaining each party's public signature or crypto key, the respective calls are not shown.

In its basic layout, the algorithm follows the registration - ballot box approach proposed by [NSS91].

Consider Figure 1. The algorithm starts with the voter filling out the ballot sheet BS , which is encrypted with a symmetric key $\left(m(BS)\right)$, which is then blinded (i.e. prepared for the blind signature[12]) $blinded\left(m(BS)\right)$. This packaged is then signed by the voter and encrypted with the public key of the registration K^{R}_{pub} and finally sent to the registration server $K^{R}_{pub}\left[S^{V}_{priv}\left(blinded(m(BS))\right)\right]$. The registration checks the identity of the voter by resolving the signature with the voter's public signature key and whether the applicant is entitled to vote. If the checks are positive, the server signs $blinded\left(m(BS)\right)$"blindly" giving $\sigma\left(blinded(m(BS))\right)$ (i.e., not knowing the encrypted ballot sheet $m(BS)$, let alone BS itself, signs with the standard private signature key S^{R}_{priv} and sends the message $S^{R}_{priv}\left(\sigma\left(blinded(m(BS))\right)\right)$ back to the voter. This message can be encrypted either by the voter's public encryption key maintained by a Trust Center, or a session key could be negotiated between voter and registration beforehand.

---

[12] The blind signature model was developed by David Chaum in 1982 [Chau82]. In general language it can be compared with the signature on a blue paper envelope.
(e,d) ... the server's blind signature pair according to the RSA system
$blinded\left(m(BS)\right) = r^{e} m(BS) \bmod(n)$ with r as random
$\sigma\left(blinded(m(BS))\right) = \left(r^{e} m(BS)\right)^{d}$ which is then divided by r giving $\frac{r^{ed}\left(m(BS)\right)^{d}}{r} = \left(m(BS)\right)^{d}$

The voter first authenticates the registration's digital signature S^R_{priv} at the latter's Trust Center and then removes the blinding layer from the signature obtaining $\sigma(m(BS))$. The voter obtains a pair of $m(BS), \sigma(m(BS))$ authenticated by the registration.

This authenticated ballot sheet is sent to the ballot box server, which checks the private signature of the registration server. In the original protocol the encrypted ballot sheet $m(BS)$ is written on a list that is published after the end of the election. The voter then checks if the encrypted vote is on the list and sends the symmetric decryption key m' to the ballot box server. The server uses this key to decrypt the ballot sheet and to count the vote. Finally, after the end of the count the keys and the ballot sheets are added to the public list so every voter can check the authenticity of the election and that it was not manipulated. Hence registration and submission of vote are done in one phase, most implementations also integrate the last phase of sending m' into the main registration/submission phase.

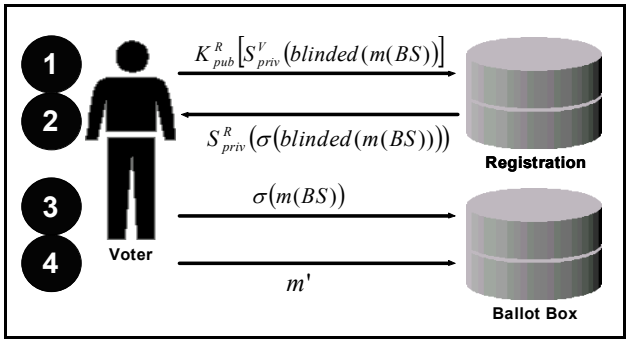


Figure 1: process model for [FOO93]

This algorithm has been implemented in various variations but all variants still maintained the basic problem: it is a one-phased algorithm, which means that both steps, identification and voting, are completed in one stage. When the administration of the registration and ballot box servers collude, it is possible to break the anonymity as well as to vote for voters that were entitled to vote but did not do so. The algorithm is secure on the application level, however, if the browser-based application (e.g., a Java applet) provided by the registration to perform the registration step fraudulently stores the IP address for each blindly signed ballot sheet, and passes on this information to the ballot box, the $m(BS)$ - and eventually also the clear-text ballot sheet after submission of m' - can be linked to a voter later. Also temporary files could be used for this purpose.[13]

Hence, anonymity cannot be guaranteed if registration and vote submission are processed in one stage.

3 PROPOSING A TWO-STAGE PROTOCOL

The proposal outlined here is a further development based on the algorithm proposed in [17].

This proposed algorithm strictly separates registration and vote submission stage following the original requirements set by [6]:

- Registration phase. The voter's credentials are checked and the voter receives a blindly signed election token, which is securely stored (see Section 4).
- Voting phase. The voter uses the election token to obtain a ballot sheet and casts her vote.

Figures 2-4 depict the proposal, in addition to the notation that was used in the last section, the voter's Trust Center T will also be used for the voting protocol.

[13] Such files could also be unintended „left-overs" from the registration process, which happened during the students union election at the University of Osnabrück in 2000 [16]

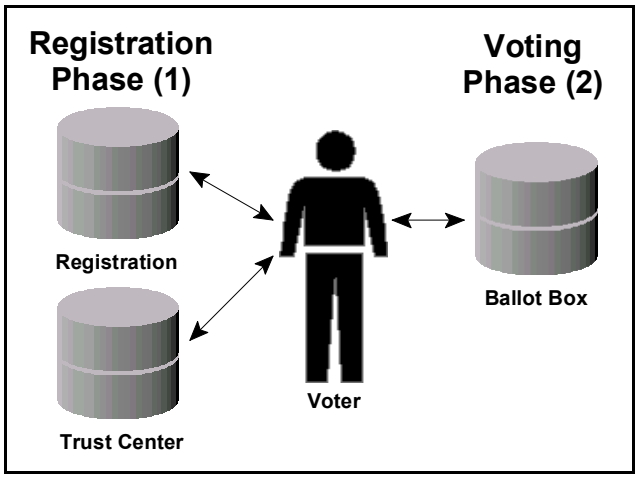


Figure 2: the two phases and the participating parties

3.1 Registration

Voters can register an arbitrary period of time before election day; since the ballot sheet is not handed out upon registration, voters can register even at a time when the list of candidates is not complete yet. As the first step the voter generates a random token t and prepares it for the blind signature, adds a text where she applies for e-voting and signs: $S^V_{priv}(blinded(t), \text{"I want to vote electronically"})$. The message is encrypted with R's public key and sent to the registration $K^R_{pub}\left[S^V_{priv}(blinded(t), \text{"I want to vote electronically"})\right]$, which verifies the voter's credentials by resolving the public signature key of the voter. If the voter is entitled to vote, the registration signs t blindly giving $\sigma_R(blinded(t))$. This is basically the same mechanism as with [13], however, since only a token is signed instead of the ballot sheet, the resulting message can be expected to be considerably shorter, which makes it easier to store the message on a secure medium.

The registration stores the electronic application and strikes the voter off the conventional voter's register. Also $\sigma_R(t)$ is stored; if the original signed token is lost and the voter re-applies for another token, the registration will always respond with the original $\sigma_R(t)$ to avoid the issue of multiple tokens.

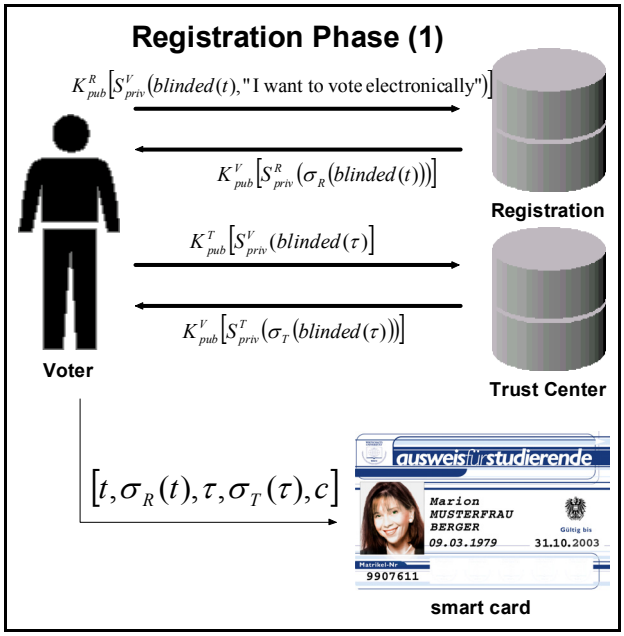


Figure 3: process model registration phase

In most elections, voters will be organized in constituencies c, this information is also sent back to the voter and has to be submitted on election day to indicate in which constituency the vote is to be counted. To avoid possible manipulation of c the blind signature keys used for $\sigma_R(t)$ can be made specific to the constituency. Hence the clear-text c submitted on election day and the authentication token issued by the registration have to point to the same c.

A similar process is repeated with the Trust Center: The voter issues a second token τ, blinds it and obtains the blindly signed $\sigma_T(\tau)$. This is required as it is the only way to make a collusion of the registration server and the ballot box server useless, as they always need the blind signature authentication of the Trust Center as well in order to forge a vote.

At the end of the registration phase, the voter holds two authentication tokens and her constituency information $[t, \sigma_R(t), \tau, \sigma_T(\tau), c]$, both tokens are needed to cast a vote on election day.

3.2 Voting

On election day the voter sends the tokens to the ballot box server to obtain a ballot sheet. This submission is not signed by the voter and the only means of authentication are the two tokens obtained earlier. The voter generates an asymmetric key m, m' to secure the communication (without disclosing the identity of the voters which would be case when using it's asymmetric crypto key pair on the signature smart card). The voter also adds the identification T of the Trust Center used, which is not used to verify the voter's identity or to obtain any public crypto key, but to choose the right Trust Center key for resolving the blind signature. The message $[c, T, m, t, \sigma_R(t), \tau, \sigma_T(\tau)]$ is encrypted with K^B_{pub} and sent to the ballot box. After decrypting the ballot box resolves the signatures $\sigma_R(t)$ and $\sigma_T(\tau)$ and if the tokens can be authenticated the ballot box issues an empty ballot sheet and encodes it with the symmetric key $(m(BS))$. The voter receives and decrypts it with m' and fills out the ballot sheet. This is then combined with the tokens, c and T, encrypted again with the public crypto key of the ballot box and sent. After authentication of the tokens, the ballot box server stores the ballot sheet and the other information received from the voter.

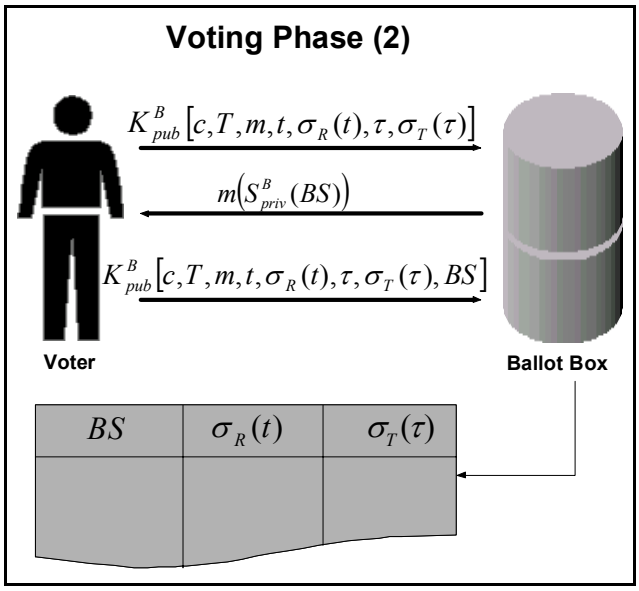


Figure 4: process model voting phase

Apart from the fact that anonymity can be guaranteed to the voter, if he uses different terminals (IP addresses) for registration and submission of vote, the server administration of the registration and the ballot box collude, votes cannot be forged, as a valid vote also has to be authenticated by a Trust Center.

3.3 Storage Media

As the algorithm uses a two-phase-protocol there is the need to temporarily store the token on a secure, anonymous medium. At this point we see three possible ways to store this token:

a) On the smart card used for the digital signature. The advantage of storing the token on the voter's smart card is the protection from data loss as compared to conventional storage media and the protection from unauthorized access when the token is secured by a PIN from reading. This method has two main problems:

- Firstly the legal provisions governing digital signatures have to allow write access at least to some area of the smart card's storage medium and the respective service provider needs to actually prepare the storage area for the storage of a token and the smart card needs to be certified for such a storage process.

- To ensure anonymity when sending the token to the ballot box there must not be any freely accessible data on the smart card identifying the card holder in a storage area which is not secured by a PIN.

The second requirement constitutes a serious problem when implementing electronic voting.

Typically, the card holder's name and other personal data is stored on the card for free read access which means that any application accessing the card can read them without any restriction. Also, every smart card has an identification number that is unique worldwide. This number is initialized during production and it enables tracing the card (and thereby the card owner) whenever it is used.

Of course, the source code of the e-voting software can be made generally available and can be submitted to certification by an independent authority showing that neither the personal data nor the card number is accessed by the voting software, however, (i) it seems doubtful whether this will be sufficient to gain public acceptance and (ii) since the election token resides on the card between registration and election day, any other application accessing the card may read the personal information plus the token stored on the card thereby enabling a third party to trace the vote later.

This shows that the common processor-smart card (as well as the whole PKI/TrustCenter-infrastructure) was designed for a completely different paradigm: Obviously it was assumed that there is no legitimate anonymous application for such a card.

Hence, smart cards used for digital signature can only be used to store the token after a fundamental redesign giving the cardholder full control as to which data is actually read from the card.

b) On any storage medium similar to an electronic purse. This variant solves the problems with serial number and clear text information discussed above: The voter uses a floppy disk or an USB-memory-key during the registration process and the token is saved on it. The implementation would be easy and would rely on general-purpose infrastructure which is available off the shelve.

The disadvantage of this solution is the relative unreliability of the medium. Especially disks are susceptible to read/write errors and the data would have to be secured manually by passwords. These media also allow unrestricted copying; even a regular backup-process can lead to multiple copies of the token. The proposed algorithm is secured against multiple use of one token, however, free duplication of the tokens is not intended – it is easy to imagine that a token that is saved on disk without any safeguard (i.e. with a PIN or password) can be found-out and can be used to undermine the voter's anonymity later.

c) On a smart card other than the smart card used for digital signature. Another possibility would also be the use of a processor smart card, whose serial number is not registered (or safeguarded by a PIN) or a storage card with a minimum of processor functionality (for an introduction cf. [18]). Pure storage cards can be read and written to by general purpose card readers and in both variations there is no need for additional hardware.

In both cases, the card used for the digital signature is used only for identification purposes during the registration phase only and the token is stored on the second card. During the voting phase, only the storage card is used and anonymity can be preserved.

The disadvantages of this method is that handling becomes more complicated (the cards need to be changed during the registration phase), the additional costs of a second card and possible security leakages due to the fact that the card holding the token is not associated with a particular person, as was the case with the signature smart card.

4 RESUME

The paper proposed a protocol for secure e-voting via the Internet. The main technical problem in the implementation is the inability of current smart cards used for digital signature to hold an election token that can be used anonymously on election day. We believe that the best solution for enabling the use of smart cards for e-voting will be to protect the user-specific information on the card (particularly the certificate and the card ID) from unauthorized access. This could be implemented by a PIN; however, since several applications can be expected to store confidential information on the card, this may lead to a large number of PINs to be remembered by the card holder (confidential Medicare information, various access data, or an election token). The alternative would be to use only one PIN for all such applications, however, this would lead to the rather unsatisfactory situation that an application that is granted access to one confidential information would also be granted access to all confidential data on the card – including the election token.

In our view, the solution can only be to grant card holders complete control over which data is actually read from the card by an application. In an analogy to the secure viewing applications for digital signatures (the text to be signed is securely displayed on screen), the user has to see and to acknowledge the information read from the card before any application can actually access it. Hence, fraudulent and unauthorized access of data on the card which is of no concern to an application can be prevented. This will also enable the use of smart cards for the purposes of electronic voting.

The protocol outlined in this paper is currently being implemented as a software prototype under a research grant of the City of Vienna. Implementation experience so far shows that - apart from security issues - the unique identification of voters is one of the primary questions in e-Voting. Here, digital smart cards are used for identification, the voter is identified via his/her citizen number in the Central Citizen Register (Zentrales Melderegister). This number is included in the digital certificate of the signature card, thus providing a unique link between the digital certificate, the entry in the voter register and the real person. For a report on the implementation progress, refer to [19].

5. REFERENCES

[1] R. Müller and A. Prosser, "The economic effects of the transition of the U.S. federal procurement to Electronic Commerce," *CEMS Business Review (2)*, pp. 295-308, 1998.

[2] A. Prosser and R. Müller, "Öffentliche Beschaffung mittels Electronic Commerce," *Wirtschaftsinformatik*, vol. 99, pp. 256-266, 1999.

[3] E-Procurement, *http://simap.eu.int* (as per 2002-03-15).

[4] Report on the National Workshop on Internet Voting, *http://www.internetpolicy.org/research/e_voting_report.pdf* (as per 2002-03-15).

[5] W. Dujmovits, *Auslandsösterreicherwahlrecht und Briefwahl*. Wien: Verlag Österreich, 2000.

[6] H. Nurmi, A. Salomaa, and L. Santean, "Secret Ballot Elections in Computer Networks," *Computers and Security 36 (10)*, pp. 553-560, 1991.

[7] Vote Here Gold, *http://www.soundcode.com/voteheregold.html* (as per 2002-05-25).

[8] J. Feghhi, J. Feghhi, and P. Williams, *Digital Signatures - Aplied Internet Security*. Reading: Addisson-Wesley, 1999.

[9] E. A. Fisch and G. B. White, *Secure Computers and Networks - Analysis, Design, and Implementation*. Boca Raton: CRC Press, 2000.

[10] EU Student Vote, *http://www.eusv.org* (as per 2002-05-25).

[11] Wahl zum Jugendgemeinderat Fellbach, *http://www.fellbach.de/wahlen* (as per 2001-11-20).

[12] Glossar zur Wahl 1999, *http://www.parlament.ch/dL/D/Wahlen/Wahlen99/Glossar_A_Z_d.htm* (as per 2002-05-25).

[13] A. Fujioka, T. Okamoto, and K. Ohta, "A Practical Secret Voting Scheme for Large Scale Elections," presented at Advances in Cryptology - AUSCRYPT92, Berlin, 1993.

[14] Sensus e-Voting system, *http://lorrie.cranor.org/voting/sensus* (as per 2002-05-23).

[15] Forschungsgruppe Internetwahlen, *http://www.internetwahlen.de* (as per 2002-05-23).

[16] Pressearchiv Forschungsgruppe Internetwahlen, *http://www.internetwahlen.de/internetwahlen/archiv2/archiv2.html* (as per 2002-05-23).

[17] A. Prosser and R. Müller-Török, "Electronic Voting via the Internet," presented at International Conference on Enterprise Information Systems ICEIS2001, Setùbal, 2001.

[18] H.-R. Hansen and G. Neumann, *Wirtschaftsinformatik 1*, 8 ed. Stuttgart: UTB, Lucius & Lucius, 2001.

[19] A. Prosser, R. Kofler, and R. Krimmer, "Implementing an Internet-based Voting System for Public Elections - A Project Experience," *submitted to ICEIS 2003*.

3.3 e-Voting.at: Current state of public elections over the Internet in Austria

Abstract: [14] *In times where the voter turnout in public elections in most western countries is sinking, e-Voting is considered as a possible mean to increase voter participation by making Voting more accessible. However, there are legal means and the infrastructure requirements which have to be met.*
This paper outlines the several requirements for implementing e-Voting in Austria and describes the current state of e-Voting at two Austrian organizations - the Austrian Student Union and the Federal Chamber of Commerce.

1 INTRODUCTION:

The Austrian constitution defines in article 1: "Austria is a democratic republic". Public elections are the key element of a democratic country, as the right of a republic is derived from the right of its inhabitants. These mandatory elections are also a human right as stated in article 3 of the protocol to the convention of the human rights to guarantee free and secret elections.[15]

Although these high standards for elections had already been discussed during the time of the Hapsburg monarchy in the 19th century they had only been installed partly. It took until 1918 with the foundation of the first republic of Austria for the standards of common, equal, free and secret elections with proportional representation to become reality for the Austrian population [Wela99].

Although the election procedures and laws were a point of big discussions, still very little changed and the Parliament has passed only two notable election reforms since then:

In 1970 the number of members of the parliament was increased to 183 and the constituencies were decreased down to nine.

The reform of 1992 resulted in the possibility to participate as Austrian from outside the country with a form of postal voting and the lowering of the age to vote and to be elected down to 18 respectively 19.

The first-order elections to the Austrian parliament are marked with an internationally respected very high participation rate of well above 90 percent. Only at the end of the 20th century it has decreased and reached an all-time-low with 80,4 percent.

Year	Voter Turnout	Year	Voter Turnout
1945	94,3	1975	92,9
1949	96,8	1979	92,9
1953	95,9	1983	92,6
1956	96,0	1986	90,5
1959	94,2	1990	90,5
1962	93,8	1994	91,9
1966	93,8	1995	85,9
1970	91,8	1999	80,4
1971	92,4		

Table 1: Voter turnout at elections to the Austrian national parliament from 1945-1999 [NaWa02]

Still the current reform discussions have not concentrated on how to reach the former mark of 90 percent turnout but like in the past on how certain political parties could increase there shares by either generally lowering the age to vote actively down to sixteen or introducing of postal voting also from inside Austria as well.

The Second Republic also knows a large number of second-order (and less important) elections. The reason for this is the Austrian concept of the social partnership (*Sozialpartnerschaft*) as a network of cham-

[14] This article was written by Prosser, Kofler and Krimmer for the workshop "Internet voting – present forms and future perspectives" on June 14th/15th 2002 in Marburg, Germany

[15] i.e. Switzerland hasn't ratified it so far as in one canton it is still casting the *Ständerat* publicly at the commencement of the *Landsgemeinde* [CHPa99]

bers, where each citizen is represented according to her social or work status.[16] All those chambers are characterized by a compulsory membership, elected representatives and a law with their rights and duties. How often elections are conducted varies from organization to organization and ranges from every two to five years. The stable political and social life in the Second Republic in contrast to the First Republic is attributed to this concept [KaTa00].

Whereas the voter turnout for the first-order elections is by international standards high, the organizations of the social partnership do have to consider their turnout much more seriously. It ranges between only thirty to sixty percent.

Year	Voter Turnout
1983	36,25
1985	29,79
1987	34,68
1989	30,1
1991	30,56
1993	30,42
1995	29,31
1997	27,6
1999	27,53
2001	28,52

Table 2: The voter turnout for the Austrian Student Union elections from 1983 – 2001 [ÖH01]

Year	Voter Turnout
1985	70,0
1990	61,9
1995	54,74
2000	59,27

Table 3: The voter turnout for the National Chamber of Commerce elections from 1985 to 2000 [Karl01]

Due to this fact, two chambers – the National Student Union and the Federal Chamber of Commerce – started in 1999 to look into electronic voting with the expectation that this new method would raise voter's participation.

In this paper we want to line out the current state of electronic voting in Austria in two steps. First we evaluate what questions determine an e-Voting system and how it is determined within the Austrian legal system and infrastructure.

We then continue with the analysis of the two projects and the experiences made so far.

2 PAPER-BASED VOTING

The current way of elections as we know has evolved over the time. Every country has its special variations based on political tradition, current legal jurisdiction and social conditions. Hence all forms can be grouped in two systems where the basic criteria are whether or not the vote is casted in a polling station or not:

- presence voting and
- distance (remote) voting

[16] Examples would be the National Student Union representing all students in Austria, the Federal Chamber of Commerce representing the employers or the Federal Chamber of Labour representing the employees.

2.1 Presence Voting

Article 26 of the Austrian Constitution[17] sets the basic rules for the elections to the national parliament. The principles of voting (*Wahlrechtsgrundsätze*) in Austria are thereby [WaMe00]:

- universal: all citizens are entitled to vote and be elected with a few exceptions explicitly named
- equal: the vote of every voter has the same impact on the election result
- immediate: voters elect the members of the parliament directly, no electoral college as intermediate elector
- personal: the voter herself has to vote (no representatives are allowed)
- secret: the public and the election administration may not get notice of the content of one's vote

Further regulations are the use of the proportional representation in contrast to the Anglo-American democracies (i.e. US, GB or New Zealand) and the requirements of the Austrian citizenship and the age of eighteen years to be eligible to vote. Another difference to the US is that Austrian citizens, residing within the country, are automatically registered to vote whereas Americans have to sign up to vote at their elections.

[17] The Austrian Constitution and any other Austrian law or finding of the Austrian constitutional court can be found in the online Law Information system of the Federal Chancellery at http://www.ris.bka.gv.at

2.2 Distance (Remote) Voting

Germany and Austria both have very similar principles of voting [GrGe00]. Still the Austrian legislative prohibited any form of remote voting until 1989. Germany had already introduced postal voting in the sixties, but the Austrian constitutional court (*Verfassungsgerichtshof*, short VfGH) declared in 1985 that this form of remote elections is in contradiction to the principles of personal and secret elections [Mars00; VfGH85]. Main problems arose around the possibility of a representative voting instead of the entitled citizen and that there is no control if the voter was supervised or coerced during the election process. In comparison to Germany, it can be said, that the Austrian constitution requires a higher protection for the voter from any undue influences that could weaken the status of the voting principles.

Not a change in the political opinion as one could assume but the court apparently changed its opinion a couple of years later, when an Austrian citizen living abroad sued the Austrian state because he was not included in the voter register of his former home town. He was therefore not allowed to vote. At that time the right to vote was linked to two facts –to the Austrian citizenship and to a residence in Austria. In its ruling in 1989 the VfGH said [VfGH89], that the entitlement to vote may not be bound to a residence in Austria and so the Austrian parliament had to change § 2 of the voter registration law (*Wählerevidenzgesetz*, short WEvG) and § 38 of the national election regulations of 1992 (*Nationalratswahlordnung*, short NRWO) to include Austrians living abroad in the national election register.[18] The Austrians remain listed in the elections register of their former residence for ten years after having left the country. After this period they may reapply to be included in the register by stating their interest in taking part in Austrian elections. For the voting procedure itself, the legislative used the two-phase concept which had already existed for the use of a voting card.

[18] For more on the Voting of Austrians living abroad see [Dujm00]

- Phase One: Some time before the election the voter (in Austria or abroad) can apply for the voting card, which exists in two versions, (i) enabling the citizen to vote in Austria but not in his/her city of residence or (ii) a voting card which can be used abroad. In the both cases the voting card consists of three parts: the ballot sheet, a neutral envelope and another envelope with the voter's name and constituency on it, so the vote can be correctly addressed. For the 2002 elections many cities and embassies accepted applications for voting cards over the Web.

- Phase Two: As Austria doesn't allow distance voting within the country, there is now a distinction between Austrians living in and outside Austria who don't vote in the city where they are registered. Austrians in Austria are required to cast their vote in any polling station and Austrians abroad are not required to have an election administration present. They can vote anywhere as long a person of equal status of an Austrian notary confirms by her signature and timestamp on the constituency envelope that the vote was filled out in privacy and independently. Following § 60 NRWO the notary can also be replaced by two Austrian citizens. Then the vote has to be mailed to the regional election administration and is only valid if it is received at the latest eight days after the Election Day.

Although there have been many initiatives to allow distance voting for regional and local elections and not only from abroad for national elections, none of them managed to get support by enough parties (a two-third majority is required to change the constitution) such as a recent by Federal Councilor Weiss [Weis99].

3 ELECTRONIC VOTING

The concept of electronic voting can be implemented with both presence and distance voting concepts as it basically is the use of computers for the election processes of (i) voter identification, (ii) vote casting and (iii) ballot counting.

Nevertheless e-Voting as it is understood in this paper se is considered as means of remote voting using the Internet. Any remote e-Voting system that is to be used for public elections following common principles of elections (as depicted in 2.1) implies that it fulfills all of the following considerations:

- identification: one must know WHO voted,
- anonymity: but neither must one know WHAT the voter has voted
- administration fraud: nor must the election administration have the ability to fake the results

3.1 Legal issues

The Austrian jurisdiction has been very strict about remote voting for national, regional or local public elections as mentioned in 2.2. But in [VfGH96] the VfGH leveraged its view on the requirements for elections of the social partners and made it possible for the national parliament in 2001 to change the Student Union law (*Hochschülerschaftsgesetz*, short HSG*)* and the Federal Chamber of Commerce law (*Wirtschaftskammergesetz,* short WKG) to allow electronic voting.

Besides the common voting principles that requires an election of ÖH and the chamber of Commerce to be equal, personal and secret, paragraph §34 of the HSG and §73 of the WKG address the problem areas described above more in detail by requiring

- the voter to use a qualified digital signature on a smart card in accordance to the Digital Signature law (*Signaturgesetz,* short Sig-G*)* to identify uniquely,

- the e-Voting system to keep the vote secret in accordance to the Data Protection Law (*Datenschutzgesetz 2000,* short DSG2000) in §4 to secure the anonymity

- and finally to check the system against the possibility of fraudulent manipulations by the election administration by requiring an appraisal of non-fraud ability by A-SIT[19] and a permission by the commission for data security following §19 DSG2000.

3.2 Infrastructure issues

The Parliament passed the Digital Signature Law in 1999 (Austria was the first country of the European Union to do so), but it took till December 2001 for the first trust center to be accredited by the national regulator [RTR01]. Furthermore the smart cards are rather expensive with a price of around 50 € and it thereby remains doubtful whether these cards will become popular without useful ("killer-") applications like e-Voting.

The strategy is now to combine the smart card functionality with already existing ID-cards like the social security card. The first larger number of smart cards being rolled out is the student ID cards of the Vienna University of Economics and Business Administration (WU). This institution issues for each of its 20.000 students a photo-ID card with the digital signature feature in fall 2002 [WUW02]. Other institutions are still in the planning stage, like the Social Security that wanted to roll out its social security card for all members residing in Vienna, but this plan had to be postponed for one more year due to technical reasons [Prof02].

Even if you use smart cards to identify the user, it is still not guaranteed that the user's credentials also comply with the citizen in the election register even though the name and date of birth match. This problem can be solved either by (i) the smart card issuing organization being the same as the election conducting body or by (ii) using a unique identifier like a citizen ID number that is stored on the smart card.

Austria chose to realize the second variant with the registration law in 1995. Hereby Austria is one of the few countries in Europe where such a central register (ZMR) for all citizens exists. It started public service on first of March 2001 and every Austrian citizen has been appointed one unique "ZMR-identification-number". However, in spite of its usefulness for e-Voting, it should be noted that critics like the data security specialists from ARGEdaten warned from misuse and called the system the first step to a surveillance state [ARGE01].

With a smart card you can sign a digital document and everyone who gets this document can access a public server to verify your identity and that you signed the document. Unfortunately the standard certificates[20] on the smart card include only given and last name and the numbers of the certificate and the card itself (the later is world wide unique). Even by looking it up in the public certificate server of the certification authority (i.e. in Austria a-trust, http://www.a-trust.at) the only further information one can get, is the status of validity of the certificate and if it has been revoked.

By now storing the national citizen ID on the citizen's smart card it becomes a national ID card (*Bürgerkarte*), which can then be used to anonymously cast a vote.

---

[19] A-SIT is the organization appointed by the Chancellor to check the security of trust centers and their services required by the Digital Signature Law.

[20] for a more detailed view on smart card standards see [Hass95]

Currently this card still imposes a security problem for e-Voting as it is not an anonymous data store medium.[21]

4 CURRENT E-VOTING INITIATIVES

Currently there is no national e-Voting project articulated or planned, which is probably due to high voter turnout. In contrast, the social partners all suffer from low voter turnout and two of them are thinking about e-Voting for some years now. In the following we show the current status of their work and their experiences.

4.1 The Austrian Student Union

As described above, the Austrian Student Union (ÖH) is one member of the social partnership and has undergone a major crisis in the 1990's. Reasons for this were on one hand the low voter turnout rate with constantly around 30 % and the public discussion of the compulsory membership of every student in the union. In a strike ballot in 1991 eighty percent of the students decided to keep up the system of a Student Union with the mandatory membership for every student. This ballot finished an Austrian-wide discussion and manifested the Student Union as the only representation of students. This led to a Student Union legal reform in 1998 with which this system was introduced for all higher academic study programs in Austria and the ÖH is now a union for all Austrian students.

Every two years the elections take place and the students can participate on up to four levels related to their field of studies, faculty, university, and finally for all students the national body of representatives. To do so, the system knows two ways of electing the officials – the study program representatives are elected personally by name and for the other three levels (faculty, university and national) one votes for political parties. This makes the system quite complicated and error-prone. For the other side - the election officials – it is also hard to organize the election, as more than 100 polling stations are required and more than 500 voluntary workers have to be found, who do not belong to a political party.

These challenges led to two developments, on the one hand to automate the election processes and on the other hand to introduce remote e-Voting.

The remote e-Voting initiative started in 2000, where a cooperation agreement was settled between the ÖH, the Ministry for Education, Science and Culture and the research group Internetwahlen of the University of Osnabrück (Germany). The aim of this was to hold the ÖH elections in 2001 over the Internet using the digital signature of the WU student ID card. As outcome of this agreement the HSG was changed to accommodate remote e-Voting but finally the project failed in March 2001 because the mandatory certification process of the digital signature card couldn't be finished early enough to run tests.

Even though the remote e-Voting could not be implemented, the ÖH at WU and the university IT department developed a voter identification system to accelerate the complicated check of the identity and voter eligibility. The card was placed in the card reader and on the monitor a picture of the student was shown (to check the identity of the voter) and the relevant ballot sheets were handed to the student to be filled out by hand.

The positive experiences with the voter identification system led then to an agreement with the WU institute for information processing and information economics, department of production management to further pursue remote e-Voting. A first work agreement has been a study, whether or not the students are into e-Voting or not.

[21] See [PrKr03] for an elaboration on that problem.

The study showed that 84% of the students would prefer the use of electronic voting, with only 5% rejecting this technology. Also, the findings showed that two groups of students exist. First the group that uses frequently the facilities of the university and because of that doesn't have a real preference for or against e-Voting. The other group is strongly in favor for e-Voting as they only come to the university campus for exams and mandatory seminars and don't have a strong attachment to the university. However, they still find it an obligation to participate in the election but the lack of time prevents them from voting at the polling station. This group will grow at the WU, as space is limited and the number of students is still growing. With new services such as the e-learning platform http://learn.wu-wien.ac.at the necessity to attend the main facilities will diminish and with that comes the danger of a further voter turnout drop if the election process is not available online as well.

As a further step, an e-Voting prototype fulfilling the legal requirements as depicted above is currently under development with the support of the City of Vienna and first results are expected for the beginning of 2003 and will be presented to the public at the e-Gov day 2003 in Vienna.

4.2 Federal Chamber of Commerce

The Federal Chamber of Commerce is in a similar situation like the student union although their voter turnout has never been comparably low but still critical with just above fifty percent in 1995. And like ÖH, their analysis led to a similar two-way strategy. First accelerating the election processes and at the same time developing a remote e-Voting strategy for deployment later on.

For the elections in 2000 the Chamber of Commerce of Vienna developed a system to connect all 64 polling stations. This enabled the election administration to automate the voter identification by offering one central electronic database with the voter registration list in replacement for the thoUSnds of paper pages with the list on it. The voter herself was able to use any one of the different stations to cast her vote in the regular way on the paper ballot sheet. For the counting process the administration again used a central scanner unit with OCR to automate the vote count and the publication of the results. They do not want to use e-Voting machines for the vote casting process due to the high development cost. In the opinion of the Chamber of Commerce this money is rather to be invested in the remote e-Voting project as described by Schinagl and Kilches in 2000 [ScKi00]. In their paper they presented a three step plan for the introduction of remote e-Voting:

1. develop a technical and organizational concept for an election with remote e-Voting capabilities as the

2. basis for a change of the chamber of commerce law WKG and then allow

3. pilot projects to gain experience for the use of remote e-Voting in the 2005 elections

So far the steps one and two have been realized with the Parliament passing the bill in June 2001 (four months after the student union law HSG). Currently pilot projects have been discussed internally but no deployment date has been fixed or decision whether or not e-Voting will be used in 2005 has been made. More likely they will pursue a broader deployment of the networked polling stations in the City of Vienna.

5 ASSESSMENT

Remote voting as such (and e-Voting as a subset specifically) is a very conflicting topic on the Austrian national level. This is not only shown by the legal point of view depicted above but also finds its (non-) representation in the national strategy papers.

The report “Austrian Information Society” [KnGr97] published by the working group installed by the federal chancellery in 1997 was the last document containing an opinion on e-Voting. This ambitious program concentrated on how to lead Austria in the next century and has been the first ICT-strategy[22] to be centrally organized and coordinated. A broad range of relevant topics, e-Business, e-Government and even e-Democracy were discussed. The working group was very skeptic towards e-Democracy. On one hand it was seen positive that using ICT can raise the level of transparency but on the other hand the so called digital divide could result in unequal conditions in the ability to access public services for the citizens. As far as e-Voting was concerned, it was not seen as an alternative to regular elections as long as 100 % security can be guaranteed and manipulation of votes or violation of the voters secrecy can be ruled out.

Currently the Austrian ICT-strategy is dominated by the EC e-Europe initiative [eEur99] where “e-Austria in e-Europe” [AktP02] is the Austrian equivalent to it. This initiative is led by the e-Austria taskforce with Prof. Posch as the National Chief Information Officer in charge [Posc01]. But not one of those documents contains any judgment or strategy towards electronic voting.

This is in contrast to the high level of attention e-Voting was given by passing laws that enable remote e-Voting as an alternative way of voting. To our knowledge no other country in the European Union did pass a national law so far that allows e-Voting with no limitation to a pilot. More interestingly so far no such pilot or public tests have taken place in Austria.

This makes it necessary to develop a national strategy for the use of e-Voting. Currently only the Austrian Computer Society (short OCG) installed a work group for e-Democracy/e-Voting[23].

[22] ICT stands for **I**nformation **C**ommunication **T**echnology

[23] More information about the Work group „e-Democracy/e-Voting“ of the OCG can be found at http://egov.ocg.at

6. REFERENCES

[AktP02] Bundeskanzerlamt: "Aktionsplan eEurope, Umsetzung in Österreich". http://www.bka.gv.at/bka/service/publikationen/infogesellschaft/apmaerz02.pdf, as available on 2002-08-15.

[ARGE01] "Überflüssiges Meldegesetz". http://www.ad.or.at/news/20010108.html, as available on 2002-08-15.

[CHPa99] "Glossar zur Wahl 1999". http://www.parlament.ch/dL/D/Wahlen/Wahlen99/Glossar_A_Z_d.htm, as available on 2002-05-25.

[Dujm00] Dujmovits, Werner: "Auslandsösterreicherwahlrecht und Briefwahl". Verlag Österreich, Wien 2000.

[eEur99] "eEurope". http://europa.eu.int/information_society/eeurope/news_library/pdf_files/initiative_en.pdf, as available on 2002-08-15.

[GrGe00] Jarass, Hans; Pieroth, Bodo: "Grundgesetz für die Bundesrepublik Deutschland". 5. Aufl., C. H. Beck'sche Verlagsbuchhandlung, München 2000.

[Hass95] Hassler, Vesna: "IT Security and Smart Card Standards". http://www.infosys.tuwien.ac.at/Staff/vh/papers/std.ps.gz, as available on 2002-08-15.

[Karl01] Karlhofer, Ferdinand: "Interessensverbände im Umbruch". In: Forum Politische Bildung (Edt.): Materialpaket Politische Bildung. Wien 2001.

[KaTa00] Karlhofer, Ferdinand; Tálos, Emmerich: "Sozialpartnerschaft unter Druck". In: Anton Pelinka; Fritz Plasser; Wolfgang Meixner (Edt.): Die Zukunft der österreichischen Demokratie. Trends, Prognosen und Szenarien. Vol. 22, Signum-Verlag, Wien 2000, p. 381-402.

[KnGr97] Knoll, Norbert; Grossendorfer, Enno: "Informationsgesellschaft". Wien: Bundeskanzleramt. 2002: 1997.

[Mars00] Marschitz, Walter: "Internet-Voting". http://www.plattform.or.at/download/POP_Art_Internetvoting.pdf, as available on 2002-08-15.

[NaWa02] "Die österreichischen Nationalratswahlen von einst bis heute". http://www.modernpolitics.at/publikationen/jahrbuch/wahlergebinsse/wahlen_index.htm, as available on 2002-09-25.

[ÖH01] Österreichische Hochschülerschaft. 2001.

[Posc01] Posch, Reinhard: "IKT-Strategie des Bundes". http://www.iaik.at/news/2001-06-25.IT-Strategie.pdf, as available on 02-08-15.

[PrKr03] Kofler, Robert; Krimmer, Robert; Prosser, Alexander: "Electronic Voting - Algorithmic and Implementation Issues". HICSS-36. Hawaii 2003

[Prof02] Palme, Liselotte: "Chip-Card-Chaos". http://www.profil.at/export/profil/p_content.php3?&xmlval_ID_KEY%5b%5d=0011&xmlval_AUSGABE[]=2002_33&mdoc_id=3883967&content=main, as available on 2002-09-09.

[RTR01] RTR: "RTR akkreditiert Datakom Austria". http://www.rtr.at/web.nsf/deutsch/Portfolio~

Presseinfos~nach%20Datum~PresseInfo Datum~PInfo181201?OpenDocument, as available on 2002-08-15.

[ScKi00] Schinagl, Wolfgang; Kilches, Ralph: "Online-Wahlen und E-Voting: Entwicklungstendenzen zu elektronischen Wirtschaftskammer-Wahlen im Jahr 2005". 3. Fakultätstag der Rechtswissenschaftlichen Fakultät. Graz 2000, S. 291-339

[VfGH85] "Erkenntnis G18/85". Verfassungsgerichtshof on 1985-03-16.

[VfGH89] "Erkenntnis G218/88". Verfassungsgerichtshof on 1989-06-13.

[VfGH96] "Erkenntnis WI-2/95". Verfassungsgerichtshof on 1995-02-29.

[WaMe00] Walter, Robert; Mayer, Heinz: "Bundesverfassungsrecht". 9. Auflage, Manz, Wien 2000.

[Weis99] Weiss, Jürgen: "Gesetzesantrag "Einführung der Briefwahl auf Landes- und Gemeindeebene"". http://www.parlinkom.at/pd/pm/XXI/I/his/000/I00005_.html, as available on 2002-08-15.

[Wela99] Welan, Manfried: "Verhältniswahlrecht - Mehrheitswahlrecht". Wien: Institut für Wirtschaft, Politik und Recht, Universität für Bodenkultur Wien. 1991.

[WUW02] "Wirtschaftsuniversität Wien". http://www.wu-wien.ac.at, as available on 2002-08-15.

4 Appendix

4.1 Systemumgebung

1 HARDWARE:

CPU	intel Pentium II – 333MHz
RAM	128 MB
m/b manufacturer:	Asus
SCSI3-subsystem	Adaptec
HD 1	IBM 3 GB
HD 2	IBM 15 GB
Network	intel PRO100S
Protokoll	Ethernet - 100Mbit

2 SOFTWARE

Betriebssystem	**Version**
SuSE Linux	8.0
Kernel:	2.4.18-4GB
Webserver	
Apache	1.3.23
Skript	
PHP	Version 4.1.0
Datenbank	
mysql	3.23.48
Hash-library	
mhash	mhash-0.8.16
Java	Java2 Runtime Environment, Standard Edition (build 1.4.1-b21) Java HotSpot(TM) Client VM (build 1.4.1-b21, mixed mode)

4.2 Datenbankumgebung

PHP Version 4.1.0

System	Linux You 2.4.17 #1 SMP Mon Dec 17 18:25:06 GMT 2001 i686 unknown
Build Date	Sep 20 2002
Configure Command	'./configure' '--prefix=/usr/share' '--datadir=/usr/share/php' '--bindir=/usr/bin' '--libdir=/usr/share' '--includedir=/usr/include' '--with-config-file-path=/etc' '--with-exec-dir=/usr/lib/php/bin' '--disable-debug' '--enable-bcmath' '--enable-calendar' '--enable-ctype' '--enable-dbase' '--enable-discard-path' '--enable-exif' '--enable-filepro' '--enable-force-cgi-redirect' '--enable-ftp' '--enable-gd-imgstrttf' '--enable-gd-native-ttf' '--enable-inline-optimization' '--enable-magic-quotes' '--enable-mbstr-enc-trans' '--enable-mbstring' '--enable-memory-limit' '--enable-safe-mode' '--enable-shmop' '--enable-sigchild' '--enable-sysvsem' '--enable-sysvshm' '--enable-track-vars' '--enable-trans-sid' '--enable-versioning' '--enable-wddx' '--enable-yp' '--with-bz2' '--with-dom=/usr/include/libxml2' '--with-ftp' '--with-gdbm' '--with-gettext' '--with-gmp' '--with-imap=yes' '--with-iodbc' '--with-jpeg-dir=/usr' '--with-ldap=yes' '--with-mcal=/usr' '--with-mcrypt' '--with-mysql=/usr' '--with-ndbm' '--with-pgsql=/usr' '--with-png-dir=/usr' '--with-qtdom=/usr/lib/qt2' '--with-snmp' '--with-t1lib' '--with-tiff-dir=/usr' '--with-ttf' '--with-freetype-dir=yes' '--with-xml' '--with-xpm-dir=/usr/X11R6' '--with-zlib=yes' '--with-openssl' '--with-curl' '--with-swf=./dist/' '--with-imap-ssl' '--with-gd=yes' '--enable-xslt' '--with-xslt-sablot' '--with-mm' '--with-apxs=/usr/sbin/apxs' 'i386-suse-linux'
Server API	Apache
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php.ini
ZEND_DEBUG	disabled
Thread Safety	disabled

This program makes use of the Zend Scripting Language Engine:
Zend Engine v1.1.0a, Copyright (c) 1998-2001 Zend Technologies

PHP 4.0 Credits

Configuration

PHP Core

Directive	Local Value	Master Value
allow_call_time_pass_reference	On	On
allow_url_fopen	1	1
always_populate_raw_post_data	0	0
arg_separator.input	&	&
arg_separator.output	&	&
asp_tags	On	On
auto_append_file	*no value*	*no value*
auto_prepend_file	*no value*	*no value*
browscap	*no value*	*no value*
default_charset	*no value*	*no value*
default_mimetype	text/html	text/html
define_syslog_variables	Off	Off
disable_functions	*no value*	*no value*
display_errors	On	On
display_startup_errors	Off	Off
doc_root	*no value*	*no value*
enable_dl	On	On
error_append_string	*no value*	*no value*
error_log	*no value*	*no value*
error_prepend_string	*no value*	*no value*
error_reporting	2039	2039
expose_php	On	On
extension_dir	/usr/share/20010901	/usr/share/20010901
file_uploads	1	1
gpc_order	GPC	GPC
highlight.bg	#FFFFFF	#FFFFFF
highlight.comment	#FF9900	#FF9900
highlight.default	#0000CC	#0000CC
highlight.html	#000000	#000000
highlight.keyword	#006600	#006600
highlight.string	#CC0000	#CC0000
html_errors	On	On
ignore_user_abort	Off	Off
implicit_flush	Off	Off
include_path	.:/usr/share/php	.:/usr/share/php
log_errors	Off	Off
magic_quotes_gpc	On	On
magic_quotes_runtime	Off	Off
magic_quotes_sybase	Off	Off
max_execution_time	30	30
memory_limit	8M	8M
open_basedir	*no value*	*no value*
output_buffering	*no value*	*no value*
output_handler	*no value*	*no value*
post_max_size	8M	8M
precision	14	14
register_argc_argv	On	On

register_globals	On	On
safe_mode	Off	Off
safe_mode_exec_dir	*no value*	*no value*
safe_mode_gid	Off	Off
safe_mode_include_dir	*no value*	*no value*
sendmail_from	me@localhost.com	me@localhost.com
sendmail_path	/usr/sbin/sendmail -t -i	/usr/sbin/sendmail -t -i
short_open_tag	On	On
SMTP	localhost	localhost
sql.safe_mode	Off	Off
track_errors	Off	Off
upload_max_filesize	2M	2M
upload_tmp_dir	*no value*	*no value*
user_dir	*no value*	*no value*
variables_order	EGPCS	EGPCS
xmlrpc_error_number	0	0
xmlrpc_errors	Off	Off
y2k_compliance	On	On

yp

YP Support	enabled

xslt

XSLT support	**enabled**

xml

XML Support	active
XML Namespace Support	active
EXPAT Version	1.95.2

wddx

WDDX Support	**enabled**
WDDX Session Serializer	enabled

swf

swf support	enabled

standard

Regex Library	Bundled library enabled
Dynamic Library Support	enabled
Path to sendmail	/usr/sbin/sendmail -t -i

Directive	Local Value	Master Value
assert.active	1	1
assert.bail	0	0
assert.callback	*no value*	*no value*
assert.quiet_eval	0	0
assert.warning	1	1
safe_mode_allowed_env_vars	PHP_	PHP_
safe_mode_protected_env_vars	LD_LIBRARY_PATH	LD_LIBRARY_PATH
session.use_trans_sid	1	1
url_rewriter.tags	a=href,area=href,frame=src, input=src,form=fakeentry	a=href,area=href,frame=src, input=src,form=fakeentry

snmp

UCD-SNMP Support	enabled
UCD-SNMP Version	4.2.3

shmop

shmop support	enabled

session

Session Support	enabled

Directive	Local Value	Master Value
session.auto_start	Off	Off
session.cache_expire	180	180
session.cache_limiter	nocache	nocache
session.cookie_domain	*no value*	*no value*
session.cookie_lifetime	0	0
session.cookie_path	/	/
session.cookie_secure	Off	Off
session.entropy_file	*no value*	*no value*
session.entropy_length	0	0
session.gc_maxlifetime	1440	1440
session.gc_probability	1	1
session.name	PHPSESSID	PHPSESSID
session.referer_check	*no value*	*no value*
session.save_handler	files	files

session.save_path	/tmp	/tmp
session.serialize_handler	php	php
session.use_cookies	On	On

posix

Revision	$Revision: 1.33.2.1 $

pgsql

PostgreSQL Support	enabled
Active Persistent Links	0
Active Links	0

Directive	Local Value	Master Value
pgsql.allow_persistent	On	On
pgsql.max_links	Unlimited	Unlimited
pgsql.max_persistent	Unlimited	Unlimited

pcre

PCRE (Perl Compatible Regular Expressions) Support	enabled
PCRE Library Version	3.4 22-Aug-2000

openssl

OpenSSL support	enabled
OpenSSL Version	OpenSSL 0.9.6c [engine] 21 dec 2001

odbc

ODBC Support	enabled
Active Persistent Links	0
Active Links	0
ODBC library	iodbc
ODBC_INCLUDE	-I/usr/local/include
ODBC_LFLAGS	-L/usr/local/lib
ODBC_LIBS	-liodbc

Directive	Local Value	Master Value
odbc.allow_persistent	On	On
odbc.check_persistent	On	On
odbc.default_db	*no value*	*no value*
odbc.default_pw	*no value*	*no value*
odbc.default_user	*no value*	*no value*
odbc.defaultbinmode	return as is	return as is
odbc.defaultlrl	return up to 4096 bytes	return up to 4096 bytes
odbc.max_links	Unlimited	Unlimited
odbc.max_persistent	Unlimited	Unlimited

mysql

MySQL Support	enabled
Active Persistent Links	0
Active Links	0
Client API version	3.23.48
MYSQL_MODULE_TYPE	external
MYSQL_SOCKET	/var/lib/mysql/mysql.sock
MYSQL_INCLUDE	-I/usr/include/mysql
MYSQL_LIBS	-L/usr/lib/mysql -lmysqlclient -L/usr/lib -z

Directive	Local Value	Master Value
mysql.allow_persistent	On	On
mysql.default_host	*no value*	*no value*
mysql.default_password	*no value*	*no value*
mysql.default_port	*no value*	*no value*
mysql.default_socket	*no value*	*no value*
mysql.default_user	*no value*	*no value*
mysql.max_links	Unlimited	Unlimited
mysql.max_persistent	Unlimited	Unlimited

mcrypt

mcrypt support	enabled
version	2.4.x
Supported ciphers	cast-128 cast-256 enigma xtea arcfour panama mars safer-sk64 saferplus des tripledes blowfish skipjack gost rc2 rc6 safer-sk128 threeway serpent idea wake loki97 rijndael-128 rijndael-192 rijndael-256 twofish blowfish-compat
Supported modes	stream cbc cfb ecb ncfb ofb nofb

Directive	Local Value	Master Value
mcrypt.algorithms_dir	*no value*	*no value*
mcrypt.modes_dir	*no value*	*no value*

mcal

MCAL Support	enabled

MCAL Version	0.6 20000121

mbstring

Multibyte (Japanese) Support	enabled
http input encoding translation	enabled

Directive	Local Value	Master Value
mbstring.detect_order	*no value*	*no value*
mbstring.http_input	*no value*	*no value*
mbstring.http_output	*no value*	*no value*
mbstring.internal_encoding	*no value*	*no value*
mbstring.substitute_character	*no value*	*no value*

ldap

LDAP Support	enabled
RCS Version	$Id: ldap.c,v 1.94.2.3 2001/12/01 14:17:43 venaas Exp $
Total Links	0/unlimited
API Version	2004

imap

IMAP Support	enabled
IMAP c-Client Version	2001
SSL Support	enabled

gmp

gmp support	enabled

gettext

GetText Support	enabled

gd

GD Support	enabled
GD Version	1.6.2 or higher
FreeType Support	enabled
FreeType Linkage	with freetype
T1Lib Support	enabled
JPG Support	enabled
PNG Support	enabled

WBMP Support	enabled

ftp

FTP support	enabled

exif

EXIF Support	enabled

domxml

DOM/XML	enabled
libxml Version	2.4.12
XPath Support	enabled
XPointer Support	enabled

dba

DBA support	enabled
Supported handlers	gdbm ndbm

curl

CURL support	enabled
CURL Information	libcurl 7.9.2 (OpenSSL 0.9.6c) (ipv6 enabled)

ctype

ctype functions	enabled (experimental)

calendar

Calendar support	enabled

bz2

BZip2 Support	Enabled
BZip2 Version	1.0.2, 30-Dec-2001

bcmath

BCMath support	enabled

zlib

ZLib Support	enabled
'zlib:' fopen wrapper	enabled
Compiled Version	1.1.3
Linked Version	1.1.3

apache

APACHE_INCLUDE	
APACHE_TARGET	
Apache Version	Apache/1.3.23
Apache Release	10323100
Apache API Version	19990320
Hostname:Port	linux.local:80
User/Group	wwwrun(30)/65534
Max Requests	Per Child: 0 - Keep Alive: on - Max Per Connection: 100
Timeouts	Connection: 300 - Keep-Alive: 15
Server Root	/usr/local/httpd
Loaded Modules	mod_userdir, mod_php4, mod_setenvif, mod_so, mod_unique_id, mod_usertrack, mod_headers, mod_expires, mod_cern_meta, mod_proxy, mod_digest, mod_auth_db, mod_auth_dbm, mod_auth_anon, mod_auth, mod_access, mod_rewrite, mod_alias, mod_speling, mod_actions, mod_imap, mod_asis, mod_cgi, mod_dir, mod_autoindex, mod_include, mod_info, mod_status, mod_negotiation, mod_mime, mod_mime_magic, mod_log_referer, mod_log_agent, mod_log_config, mod_define, mod_env, mod_vhost_alias, mod_mmap_static, http_core

Directive	Local Value	Master Value
child_terminate	0	0
engine	1	1
last_modified	0	0
xbithack	0	0

Apache Environment

Variable	Value
DOCUMENT_ROOT	/usr/local/httpd/htdocs
HTTP_ACCEPT	text/xml,application/xml,application/xhtml+xml,text/html;q=0.9, text/plain;q=0.8,video/x-mng,image/png,image/jpeg,image/gif;q=0.2,text/css,*/*;q=0.
HTTP_ACCEPT_CHARSET	ISO-8859-15, utf-8;q=0.66, *;q=0.66
HTTP_ACCEPT_ENCODING	gzip, deflate, compress;q=0.9
HTTP_ACCEPT_LANGUAGE	de-at, en;q=0.66, en-us;q=0.33
HTTP_CONNECTION	keep-alive
HTTP_HOST	137.208.42.18
HTTP_KEEP_ALIVE	300

HTTP_USER_AGENT	Mozilla/5.0 (Windows; U; Windows NT 5.1; de-AT; rv:1.1) Gecko/20020826
PATH	/sbin:/usr/sbin:/usr/local/sbin:/root/bin:/usr/local/bin:/usr/bin:/usr/X11R6/bin: /bin:/usr/games:/opt/gnome/bin:/opt/kde3/bin:/opt/kde2/bin:/opt/gnome/bin
REMOTE_ADDR	137.208.42.17
REMOTE_PORT	1215
SCRIPT_FILENAME	/home/evote/www/KoflerSkripte/phpinfo.php
SERVER_ADDR	137.208.42.18
SERVER_ADMIN	root@linux.local
SERVER_NAME	linux.local
SERVER_PORT	80
SERVER_SIGNATURE	<ADDRESS>Apache/1.3.23 Server at linux.local Port 80</ADDRESS>
SERVER_SOFTWARE	Apache/1.3.23 (Unix) PHP/4.1.0
UNIQUE_ID	Pcv9OYnQKhIAAAk0Bvg
GATEWAY_INTERFACE	CGI/1.1
SERVER_PROTOCOL	HTTP/1.1
REQUEST_METHOD	GET
QUERY_STRING	
REQUEST_URI	/evote/KoflerSkripte/phpinfo.php
SCRIPT_NAME	/evote/KoflerSkripte/phpinfo.php

HTTP Headers Information

HTTP Request Headers	
HTTP Request	GET /evote/KoflerSkripte/phpinfo.php HTTP/1.1
Accept	text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,video/x-mng,imag
Accept-Charset	ISO-8859-15, utf-8;q=0.66, *;q=0.66
Accept-Encoding	gzip, deflate, compress;q=0.9
Accept-Language	de-at, en;q=0.66, en-us;q=0.33
Connection	keep-alive
Host	137.208.42.18
Keep-Alive	300
User-Agent	Mozilla/5.0 (Windows; U; Windows NT 5.1; de-AT; rv:1.1) Gecko/20020826
HTTP Response Headers	
X-Powered-By	PHP/4.1.0
Keep-Alive	timeout=15, max=100
Connection	Keep-Alive
Transfer-Encoding	chunked
Content-Type	text/html

Additional Modules

sysvshm
sysvsem
filepro

dbase
session mm

Environment

Variable	Value
PWD	/root
DBROOT	/dev/null
PAGER	less
HOSTNAME	linux
LD_LIBRARY_PATH	:/lib:/lib
LESSCLOSE	lessclose.sh %s %s
RC_LANG	de_DE@euro
LS_OPTIONS	-a -N --color=tty -T 0
QTDIR	/usr/lib/qt3
OPENWINHOME	/usr/openwin
_SUSECONFIG_PROFILE	true
LESSKEY	/etc/lesskey.bin
LESSOPEN	lessopen.sh %s
MANPATH	/usr/share/man:/usr/local/man:/usr/X11R6/man:/opt/gnome/man:/usr/openv
GNOME_PATH	:/opt/gnome:/usr
NNTPSERVER	news
LESS	-M -I
USER	root
MACHTYPE	i686-suse-linux
XKEYSYMDB	/usr/X11R6/lib/X11/XKeysymDB
RC_LC_COLLATE	POSIX
MAIL	/var/mail/root
INPUTRC	/etc/inputrc
LINES	49
LESS_ADVANCED_PREPROCESSOR	no
GNOMEDIR	/opt/gnome
HOST	linux
COLORTERM	1
ORACLE_HOME	
INFOPATH	/usr/local/info:/usr/share/info:/usr/info
DISPLAY	137.208.42.17:0.0
LOGNAME	root
SHLVL	3
TEXINPUTS	:/root/.TeX:/usr/share/doc/.TeX:/usr/doc/.TeX
LC_CTYPE	de_DE@euro
COLUMNS	133
MINICOM	-c on

INFODIR	/usr/local/info:/usr/share/info:/usr/info
SHELL	/bin/bash
PRINTER	lp
HOSTTYPE	i386
CPU	i686
OSTYPE	linux
WINDOWMANAGER	/usr/X11R6/bin/kde
HOME	/root
TERM	xterm
XNLSPATH	/usr/X11R6/lib/X11/nls
no_proxy	localhost
PATH	/sbin:/usr/sbin:/usr/local/sbin:/root/bin:/usr/local/bin:/usr/bin:/usr/X11R6/bin: /bin:/usr/games:/opt/gnome/bin:/opt/kde3/bin:/opt/kde2/bin:/opt/gnome/bin
SSH_TTY	/dev/pts/1
mc	() { mkdir -p $HOME/.mc/tmp 2>/dev/null; chmod 700 $HOME/.mc/tmp; MC=$HOME/.mc/tmp/mc-$$; /usr/bin/mc -P "$@" >"$MC"; cd "`cat $MC`"; rm -f "$MC"; unset MC }
_	/sbin/startproc
PREVLEVEL	N
RUNLEVEL	5
DAEMON	/usr/sbin/httpd

PHP Variables

Variable	Value
PHP_SELF	/evote/KoflerSkripte/phpinfo.php
_SERVER["DOCUMENT_ROOT"]	/usr/local/httpd/htdocs
_SERVER["HTTP_ACCEPT_CHARSET"]	ISO-8859-15, utf-8;q=0.66, *;q=0.66
_SERVER["HTTP_ACCEPT_ENCODING"]	gzip, deflate, compress;q=0.9
_SERVER["HTTP_ACCEPT_LANGUAGE"]	de-at, en;q=0.66, en-us;q=0.33
_SERVER["HTTP_CONNECTION"]	keep-alive
_SERVER["HTTP_HOST"]	137.208.42.18
_SERVER["HTTP_KEEP_ALIVE"]	300
_SERVER["HTTP_USER_AGENT"]	Mozilla/5.0 (Windows; U; Windows NT 5.1; de-AT; rv:1.1) Gecko/
_SERVER["REMOTE_ADDR"]	137.208.42.17
_SERVER["REMOTE_PORT"]	1215
_SERVER["SCRIPT_FILENAME"]	/home/evote/www/KoflerSkripte/phpinfo.php
_SERVER["SERVER_ADDR"]	137.208.42.18
_SERVER["SERVER_ADMIN"]	root@linux.local
_SERVER["SERVER_NAME"]	linux.local
_SERVER["SERVER_PORT"]	80

_SERVER["SERVER_SIGNATURE"]	<ADDRESS>Apache/1.3.23 Server at linux.local Port 80</ADDR
_SERVER["SERVER_SOFTWARE"]	Apache/1.3.23 (Unix) PHP/4.1.0
_SERVER["UNIQUE_ID"]	Pcv9OYnQKhIAAAk0Bvg
_SERVER["GATEWAY_INTERFACE"]	CGI/1.1
_SERVER["SERVER_PROTOCOL"]	HTTP/1.1
_SERVER["REQUEST_METHOD"]	GET
_SERVER["QUERY_STRING"]	
_SERVER["REQUEST_URI"]	/evote/KoflerSkripte/phpinfo.php
_SERVER["SCRIPT_NAME"]	/evote/KoflerSkripte/phpinfo.php
_SERVER["PATH_TRANSLATED"]	/home/evote/www/KoflerSkripte/phpinfo.php
_SERVER["PHP_SELF"]	/evote/KoflerSkripte/phpinfo.php
_SERVER["argv"]	Array ( )
_SERVER["argc"]	0
_ENV["PWD"]	/root
_ENV["DBROOT"]	/dev/null
_ENV["PAGER"]	less
_ENV["HOSTNAME"]	linux
_ENV["LD_LIBRARY_PATH"]	:/lib:/lib
_ENV["LESSCLOSE"]	lessclose.sh %s %s
_ENV["RC_LANG"]	de_DE@euro
_ENV["LS_OPTIONS"]	-a -N --color=tty -T 0
_ENV["QTDIR"]	/usr/lib/qt3
_ENV["OPENWINHOME"]	/usr/openwin
_ENV["_SUSECONFIG_PROFILE"]	true
_ENV["LESSKEY"]	/etc/lesskey.bin
_ENV["LESSOPEN"]	lessopen.sh %s
_ENV["MANPATH"]	/usr/share/man:/usr/local/man:/usr/X11R6/man:/opt/gnome/man:/
_ENV["GNOME_PATH"]	:/opt/gnome:/usr
_ENV["NNTPSERVER"]	news
_ENV["LESS"]	-M -I
_ENV["USER"]	root
_ENV["MACHTYPE"]	i686-suse-linux
_ENV["XKEYSYMDB"]	/usr/X11R6/lib/X11/XKeysymDB
_ENV["RC_LC_COLLATE"]	POSIX
_ENV["MAIL"]	/var/mail/root
_ENV["INPUTRC"]	/etc/inputrc
_ENV["LINES"]	49
_ENV["LESS_ADVANCED_PREPROCESSOR"]	no
_ENV["GNOMEDIR"]	/opt/gnome
_ENV["HOST"]	linux

_ENV["COLORTERM"]	1
_ENV["ORACLE_HOME"]	
_ENV["INFOPATH"]	/usr/local/info:/usr/share/info:/usr/info
_ENV["DISPLAY"]	137.208.42.17:0.0
_ENV["LOGNAME"]	root
_ENV["SHLVL"]	3
_ENV["TEXINPUTS"]	:/root/.TeX:/usr/share/doc/.TeX:/usr/doc/.TeX
_ENV["LC_CTYPE"]	de_DE@euro
_ENV["COLUMNS"]	133
_ENV["MINICOM"]	-c on
_ENV["INFODIR"]	/usr/local/info:/usr/share/info:/usr/info
_ENV["SHELL"]	/bin/bash
_ENV["PRINTER"]	lp
_ENV["HOSTTYPE"]	i386
_ENV["CPU"]	i686
_ENV["OSTYPE"]	linux
_ENV["WINDOWMANAGER"]	/usr/X11R6/bin/kde
_ENV["HOME"]	/root
_ENV["TERM"]	xterm
_ENV["XNLSPATH"]	/usr/X11R6/lib/X11/nls
_ENV["no_proxy"]	localhost
_ENV["PATH"]	/sbin:/usr/sbin:/usr/local/sbin:/root/bin:/usr/local/bin:/usr/bin:/usr/X /bin:/usr/games:/opt/gnome/bin:/opt/kde3/bin:/opt/kde2/bin:/opt/gr
_ENV["SSH_TTY"]	/dev/pts/1
_ENV["mc"]	() { mkdir -p $HOME/.mc/tmp 2>/dev/null; chmod 700 $HOME/.mc/tmp; MC=$HOME/.mc/tmp/mc-$$; /usr/bin/mc -P \"$@\" >\"$MC\"; cd \"`cat $MC`\"; rm -f \"$MC\"; unset MC }
ENV[""]	/sbin/startproc
_ENV["PREVLEVEL"]	N
_ENV["RUNLEVEL"]	5
_ENV["DAEMON"]	/usr/sbin/httpd

PHP License

4.3 Java Applet Dokumentation

Package **Class** Tree Deprecated Index Help

PREV CLASS NEXT CLASS | FRAMES NO FRAMES

SUMMARY: NESTED | FIELD | CONSTR | METHOD | DETAIL: FIELD | CONSTR | METHOD

Class B2Parser

```
java.lang.Object
  |
  +--B2Parser
```

public class **B2Parser**
extends java.lang.Object

This class parses the server reply which is enclosed in brackets like `[VAR][v1]xxx[/v1][v2]yyy[/v2][/VAR]`.

```
Usage:
     B2Parser b2 = new B2Parser();
     b2.setInputString(toParse);
     b2.parse();
     java.util.Properties p = b2.getOutputProperties();
     java.util.LinkedList keys = b2.getOutputLLKey();
     java.util.LinkedList values = b2.getOutputLLValue();
```

The String `[VAR][v1]xxx[/v1][v2]yyy[/v2][/VAR]` parsed with this class would yield these results:

```
     Properties p: v1=xxx, v2=yyy
     LinkedList keys: v1, v2
     LinkedList values: xxx, yyy
```

Author: Martin Karl Unger
Last update: 2002-08-17

Constructor Summary

B2Parser() The default constructor sets the *open bracket char* and the *close bracket char* and the *slash char* to their default values `[]/`.
B2Parser(java.lang.String toParse) Calls the default constructor and supplies the String which will be parsed.
B2Parser(java.lang.String toParse, char openBracketCharacter, char closeBracketCharacter, char slashCharacter) Supplies the String which will be parsed and allows to use other special characters than `[]/`.
B2Parser(java.lang.String toParse, char openBracketCharacter, char closeBracketCharacter, char slashCharacter, java.util.Properties outProperties, java.util.LinkedList outLLKey, java.util.LinkedList outLLValue)
B2Parser(java.lang.String toParse, java.util.Properties outProperties, java.util.LinkedList outLLKey, java.util.LinkedList outLLValue) Calls the default constructor and supplies the argument and sets references for the three output fields

Method Summary

char	**getCbc**() Returns the close bracket character.
java.lang.String	**getInputString**() Echoes the argument of setInputString()
char	**getObc**() Returns the open bracket character.
java.util.LinkedList	**getOutputLLKey**() Returns the keys in outputLLKey.
java.util.LinkedList	**getOutputLLValue**() Returns the values in outputLLValue.
java.util.Properties	**getOutputProperties**() Returns the outputProperties.
char	**getSlc**() Returns the slash character.
void	**parse**() Does the main work.
void	**setCbC**(char closeBracketCharacter) Sets the close bracket character (default is]).
void	**setInputString**(java.lang.String toParse) Sets the input string which will be parsed.
void	**setObc**(char openBracketCharacter) Sets the open bracket character, (default is [ ).
void	**setOutputLLKey**(java.util.LinkedList key) Sets a reference for outputLLKey.
void	**setOutputLLValue**(java.util.LinkedList value) Sets a reference for outputLLValue.
void	**setOutputProperties**(java.util.Properties p) Sets a reference for the outputProperties.
void	**setSlc**(char slashCharacter) Sets the slash character (default is /).

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

B2Parser

```
public B2Parser()
```

The default constructor sets the *open bracket char* and the *close bracket char* and the *slash char* to their default values `[]/`.

B2Parser

```
public B2Parser(java.lang.String toParse)
```

Calls the default constructor and supplies the String which will be parsed.

B2Parser

```
public B2Parser(java.lang.String toParse,
                char openBracketCharacter,
                char closeBracketCharacter,
                char slashCharacter)
```

Supplies the String which will be parsed and allows to use other special characters than `[]/`.

B2Parser

```
public B2Parser(java.lang.String toParse,
                java.util.Properties outProperties,
                java.util.LinkedList outLLKey,
                java.util.LinkedList outLLValue)
```

Calls the default constructor and supplies the argument and sets references for the three output fields

---

B2Parser

```
public B2Parser(java.lang.String toParse,
                char openBracketCharacter,
                char closeBracketCharacter,
                char slashCharacter,
                java.util.Properties outProperties,
                java.util.LinkedList outLLKey,
                java.util.LinkedList outLLValue)
```

Method Detail

setInputString

```
public void setInputString(java.lang.String toParse)
```

Sets the input string which will be parsed.

---

getInputString

```
public java.lang.String getInputString()
```

Echoes the argument of setInputString()

---

getOutputProperties

```
public java.util.Properties getOutputProperties()
```

Returns the outputProperties.

---

setOutputProperties

```
public void setOutputProperties(java.util.Properties p)
```

Sets a reference for the outputProperties.

---

getOutputLLKey

```
public java.util.LinkedList getOutputLLKey()
```

Returns the keys in outputLLKey.

---

setOutputLLKey

```
public void setOutputLLKey(java.util.LinkedList key)
```

Sets a reference for outputLLKey.

---

getOutputLLValue

```
public java.util.LinkedList getOutputLLValue()
```

Returns the values in outputLLValue.

---

setOutputLLValue

```
public void setOutputLLValue(java.util.LinkedList value)
```

Sets a reference for outputLLValue.

---

setObc

```
public void setObc(char openBracketCharacter)
```

Sets the open bracket character, (default is [ ).

---

setCbC

```
public void setCbC(char closeBracketCharacter)
```

Sets the close bracket character (default is]).

---

setSlc

```
public void setSlc(char slashCharacter)
```

Sets the slash character (default is /).

---

getObc

```
public char getObc()
```

Returns the open bracket character.

---

getCbc

```
public char getCbc()
```

Returns the close bracket character.

---

getSlc

```
public char getSlc()
```

Returns the slash character.

---

parse

```
public void parse()
          throws java.lang.IllegalArgumentException
```

Does the main work. Takes as input the member fields inputString, obc, cbc, slc and writes as output the member fields outputProperties, outputLLKey, outputLLValue.

```
java.lang.IllegalArgumentException
```

Package **Class** Tree Deprecated Index Help

PREV CLASS NEXT CLASS FRAMES NO FRAMES

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

Class B2ParserTest

```
java.lang.Object
  |
  +--java.awt.Component
        |
        +--java.awt.Container
              |
              +--java.awt.Panel
                    |
                    +--java.applet.Applet
                          |
                          +--B2ParserTest
```

All Implemented Interfaces:
javax.accessibility.Accessible, java.awt.event.ActionListener, java.util.EventListener, java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable

public class **B2ParserTest**
extends java.applet.Applet
implements java.io.Serializable, java.awt.event.ActionListener

This applet tests my class B2Parser.
Author: Martin Karl Unger
Last update: 2002-08-17

See Also:
Serialized Form

Nested Class Summary

Nested classes inherited from class java.applet.Applet
java.applet.Applet.AccessibleApplet

Nested classes inherited from class java.awt.Panel
java.awt.Panel.AccessibleAWTPanel

Nested classes inherited from class java.awt.Container
java.awt.Container.AccessibleAWTContainer

Nested classes inherited from class java.awt.Component
java.awt.Component.AccessibleAWTComponent, java.awt.Component.BltBufferStrategy, java.awt.Component.FlipBufferStrategy

Field Summary

B2Parser	**b2**
java.lang.String	**helpText**
java.lang.String	**testData**

Fields inherited from class java.awt.Component
BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT, TOP_ALIGNMENT

Fields inherited from interface java.awt.image.ImageObserver
ABORT, ALLBITS, ERROR, FRAMEBITS, HEIGHT, PROPERTIES, SOMEBITS, WIDTH

Constructor Summary

B2ParserTest()

Method Summary

void	**actionPerformed**(java.awt.event.ActionEvent evt) Here the ActionListener is implemented.
void	**actionPStart**(java.awt.event.ActionEvent evt) This method is called by actionPerformed() and will parse the text in the TextArea.
void	**init**()

Methods inherited from class java.applet.Applet
destroy, getAccessibleContext, getAppletContext, getAppletInfo, getAudioClip, getAudioClip, getCodeBase, getDocumentBase, getImage, getImage, getLocale, getParameter, getParameterInfo, isActive, newAudioClip, play, play, resize, resize, setStub, showStatus, start, stop

Methods inherited from class java.awt.Panel
addNotify

Methods inherited from class java.awt.Container
add, add, add, add, add, addContainerListener, addImpl, addPropertyChangeListener, addPropertyChangeListener, applyComponentOrientation, areFocusTraversalKeysSet, countComponents, deliverEvent, doLayout, findComponentAt, findComponentAt, getAlignmentX,

getAlignmentY, getComponent, getComponentAt, getComponentAt, getComponentCount, getComponents, getContainerListeners, getFocusTraversalKeys, getFocusTraversalPolicy, getInsets, getLayout, getListeners, getMaximumSize, getMinimumSize, getPreferredSize, insets, invalidate, isAncestorOf, isFocusCycleRoot, isFocusCycleRoot, isFocusTraversalPolicySet, layout, list, list, locate, minimumSize, paint, paintComponents, paramString, preferredSize, print, printComponents, processContainerEvent, processEvent, remove, remove, removeAll, removeContainerListener, removeNotify, setFocusCycleRoot, setFocusTraversalKeys, setFocusTraversalPolicy, setFont, setLayout, transferFocusBackward, transferFocusDownCycle, update, validate, validateTree

Methods inherited from class java.awt.Component

action, add, addComponentListener, addFocusListener, addHierarchyBoundsListener, addHierarchyListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, addMouseWheelListener, bounds, checkImage, checkImage, coalesceEvents, contains, contains, createImage, createImage, createVolatileImage, createVolatileImage, disable, disableEvents, dispatchEvent, enable, enable, enableEvents, enableInputMethods, firePropertyChange, firePropertyChange, firePropertyChange, getBackground, getBounds, getBounds, getColorModel, getComponentListeners, getComponentOrientation, getCursor, getDropTarget, getFocusCycleRootAncestor, getFocusListeners, getFocusTraversalKeysEnabled, getFont, getFontMetrics, getForeground, getGraphics, getGraphicsConfiguration, getHeight, getHierarchyBoundsListeners, getHierarchyListeners, getIgnoreRepaint, getInputContext, getInputMethodListeners, getInputMethodRequests, getKeyListeners, getLocation, getLocation, getLocationOnScreen, getMouseListeners, getMouseMotionListeners, getMouseWheelListeners, getName, getParent, getPeer, getPropertyChangeListeners, getPropertyChangeListeners, getSize, getSize, getToolkit, getTreeLock, getWidth, getX, getY, gotFocus, handleEvent, hasFocus, hide, imageUpdate, inside, isBackgroundSet, isCursorSet, isDisplayable, isDoubleBuffered, isEnabled, isFocusable, isFocusOwner, isFocusTraversable, isFontSet, isForegroundSet, isLightweight, isOpaque, isShowing, isValid, isVisible, keyDown, keyUp, list, list, list, location, lostFocus, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove, mouseUp, move, nextFocus, paintAll, postEvent, prepareImage, prepareImage, printAll, processComponentEvent, processFocusEvent, processHierarchyBoundsEvent, processHierarchyEvent, processInputMethodEvent, processKeyEvent, processMouseEvent, processMouseMotionEvent, processMouseWheelEvent, remove, removeComponentListener, removeFocusListener, removeHierarchyBoundsListener, removeHierarchyListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removeMouseWheelListener, removePropertyChangeListener, removePropertyChangeListener, repaint, repaint, repaint, repaint, requestFocus, requestFocus, requestFocusInWindow, requestFocusInWindow, reshape, setBackground, setBounds, setBounds, setComponentOrientation, setCursor, setDropTarget, setEnabled, setFocusable, setFocusTraversalKeysEnabled, setForeground, setIgnoreRepaint, setLocale, setLocation, setLocation, setName, setSize, setSize, setVisible, show, show, size, toString, transferFocus, transferFocusUpCycle

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

b2

public B2Parser **b2**

helpText

public final java.lang.String **helpText**

See Also:
Constant Field Values

testData

public final java.lang.String **testData**

See Also:
Constant Field Values

Constructor Detail

B2ParserTest

public **B2ParserTest**()

Method Detail

init

public void **init**()

Overrides:
`init` in class `java.applet.Applet`

actionPerformed

public void **actionPerformed**(java.awt.event.ActionEvent evt)

Here the ActionListener is implemented.

Specified by:
`actionPerformed` in interface `java.awt.event.ActionListener`

actionPStart

public void **actionPStart**(java.awt.event.ActionEvent evt)

This method is called by actionPerformed() and will parse the text in the TextArea.

Package **Class** Tree Deprecated Index Help
PREV CLASS NEXT CLASS FRAMES NO FRAMES
SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

Class Base64

```
java.lang.Object
  |
  +--Base64
```

public class **Base64**
extends java.lang.Object

Encodes and decodes to and from Base64 notation.

Change Log:

- v1.3.6 - Fixed OutputStream.flush() so that 'position' is reset.
- v1.3.5 - Added flag to turn on and off line breaks. Fixed bug in input stream where last buffer being read, if not completely full, was not returned.
- v1.3.4 - Fixed when "improperly padded stream" error was thrown at the wrong time.
- v1.3.3 - Fixed I/O streams which were totally messed up.

I am placing this code in the Public Domain. Do with it as you will. This software comes with no guarantees or warranties but with plenty of well-wishing instead! Please visit http://iharder.net/xmlizable periodically to check for updates or to contribute improvements.

Version:
1.3.4
Author:
Robert Harder , rob@iharder.net

Nested Class Summary

static class	**Base64.InputStream** A Base64#InputStream will read data from another InputStream, given in the constructor, and encode/decode to/from Base64 notation on the fly.
static class	**Base64.OutputStream** A Base64#OutputStream will write data to another OutputStream, given in the constructor, and encode/decode to/from Base64 notation on the fly.

Field Summary

static boolean	**DECODE** Specify decoding (value is false).
static boolean	**ENCODE** Specify encoding (value is true).

Method Summary

static byte[]	**decode**(byte[] source, int off, int len) Decodes Base64 content in byte array format and returns the decoded byte array.
static byte[]	**decode**(java.lang.String s) Decodes data from Base64 notation.
static java.lang.Object	**decodeToObject**(java.lang.String encodedObject) Attempts to decode Base64 data and deserialize a Java Object within.
static java.lang.String	**decodeToString**(java.lang.String s) Decodes data from Base64 notation and returns it as a string.
static java.lang.String	**encodeBytes**(byte[] source) Encodes a byte array into Base64 notation.
static java.lang.String	**encodeBytes**(byte[] source, boolean breakLines) Encodes a byte array into Base64 notation.
static java.lang.String	**encodeBytes**(byte[] source, int off, int len) Encodes a byte array into Base64 notation.
static java.lang.String	**encodeBytes**(byte[] source, int off, int len, boolean breakLines) Encodes a byte array into Base64 notation.
static java.lang.String	**encodeObject**(java.io.Serializable serializableObject) Serializes an object and returns the Base64-encoded version of that serialized object.
static java.lang.String	**encodeObject**(java.io.Serializable serializableObject, boolean breakLines) Serializes an object and returns the Base64-encoded version of that serialized object.
static java.lang.String	**encodeString**(java.lang.String s) Encodes a string in Base64 notation with line breaks after every 75 Base64 characters.
static java.lang.String	**encodeString**(java.lang.String s, boolean breakLines) Encodes a string in Base64 notation with line breaks after every 75 Base64 characters.
static void	**main**(java.lang.String[] args) Testing.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

ENCODE

```
public static final boolean ENCODE
```

Specify encoding (value is `true`).

See Also:
Constant Field Values

DECODE

```
public static final boolean DECODE
```

Specify decoding (value is `false`).

See Also:
Constant Field Values

Method Detail

main

```
public static void main(java.lang.String[] args)
```

Testing. Feel free--in fact I encourage you--to throw out this entire "main" method when you actually deploy this code.

---

encodeObject

```
public static java.lang.String encodeObject(java.io.Serializable serializableObj
```

Serializes an object and returns the Base64-encoded version of that serialized object. If the object cannot be serialized or there is another error, the method will return `null`.

Parameters:
`serializableObject` - The object to encode
Returns:
The Base64-encoded object
Since:
1.4

---

encodeObject

```
public static java.lang.String encodeObject(java.io.Serializable serializableObj
                                            boolean breakLines)
```

Serializes an object and returns the Base64-encoded version of that serialized object. If the object cannot be serialized or there is another error, the method will return `null`.

Parameters:
`serializableObject` - The object to encode
`breakLines` - Break lines at 80 characters or less.
Returns:
The Base64-encoded object
Since:
1.4

---

encodeBytes

```
public static java.lang.String encodeBytes(byte[] source)
```

Encodes a byte array into Base64 notation. Equivalen to calling `encodeBytes( source, 0, source.length )`

Parameters:
`source` - The data to convert
Since:
1.4

---

encodeBytes

```
public static java.lang.String encodeBytes(byte[] source,
                                           boolean breakLines)
```

Encodes a byte array into Base64 notation. Equivalen to calling `encodeBytes( source, 0, source.length )`

Parameters:
`source` - The data to convert
`breakLines` - Break lines at 80 characters or less.
Since:
1.4

---

encodeBytes

```
public static java.lang.String encodeBytes(byte[] source,
                                           int off,
```

```
                                    int len)
```

Encodes a byte array into Base64 notation.

Parameters:
`source` - The data to convert
`off` - Offset in array where conversion should begin
`len` - Length of data to convert
Since:
1.4

---

encodeBytes

```
public static java.lang.String encodeBytes(byte[] source,
                                           int off,
                                           int len,
                                           boolean breakLines)
```

Encodes a byte array into Base64 notation.

Parameters:
`source` - The data to convert
`off` - Offset in array where conversion should begin
`len` - Length of data to convert
`breakLines` - Break lines at 80 characters or less.
Since:
1.4

---

encodeString

```
public static java.lang.String encodeString(java.lang.String s)
```

Encodes a string in Base64 notation with line breaks after every 75 Base64 characters.

Parameters:
`s` - the string to encode
Returns:
the encoded string
Since:
1.3

---

encodeString

```
public static java.lang.String encodeString(java.lang.String s,
                                            boolean breakLines)
```

Encodes a string in Base64 notation with line breaks after every 75 Base64 characters.

Parameters:
`s` - the string to encode
`breakLines` - Break lines at 80 characters or less.
Returns:
the encoded string
Since:
1.3

---

decode

```
public static byte[] decode(java.lang.String s)
```

Decodes data from Base64 notation.

Parameters:
`s` - the string to decode
Returns:
the decoded data
Since:
1.4

---

decodeToString

```
public static java.lang.String decodeToString(java.lang.String s)
```

Decodes data from Base64 notation and returns it as a string. Equivlaent to calling `new String( decode( s ) )`

Parameters:
`s` - the strind to decode
Returns:
The data as a string
Since:
1.4

---

decodeToObject

```
public static java.lang.Object decodeToObject(java.lang.String encodedObject)
```

Attempts to decode Base64 data and deserialize a Java Object within. Returns `null if there was an error.`

```
Parameters:
    encodedObject - The Base64 data to decode
Returns:
    The decoded and deserialized object
Since:
    1.4
```

---

decode

```
public static byte[] decode(byte[] source,
                            int off,
                            int len)
```

Decodes Base64 content in byte array format and returns the decoded byte array.

Parameters:
source - The Base64 encoded data
off - The offset of where to begin decoding
len - The length of characters to decode

Returns:
decoded data

Since:
1.3

Package **Class** Tree Deprecated Index Help

PREV CLASS NEXT CLASS FRAMES NO FRAMES

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

Class Base64.InputStream

```
java.lang.Object
  |
  +--java.io.InputStream
        |
        +--java.io.FilterInputStream
              |
              +--Base64.InputStream
```

Enclosing class:
Base64

public static class **Base64.InputStream**
extends java.io.FilterInputStream

A `Base64#InputStream` will read data from another `InputStream`, given in the constructor, and encode/decode to/from Base64 notation on the fly.

Since:
1.3
See Also:
`Base64`, `FilterInputStream`

Field Summary

Fields inherited from class java.io.FilterInputStream
`in`

Constructor Summary

Base64.InputStream(java.io.InputStream in)	Constructs a `Base64#InputStream` in DECODE mode.
Base64.InputStream(java.io.InputStream in, boolean encode)	Constructs a `Base64#InputStream` in either ENCODE or DECODE mode.
Base64.InputStream(java.io.InputStream in, boolean encode, boolean breakLines)	Constructs a `Base64#InputStream` in either ENCODE or DECODE mode.

Method Summary

int	**read**() Reads enough of the input stream to convert to/from Base64 and returns the next byte.
int	**read**(byte[] dest, int off, int len) Calls `read()` repeatedly until the end of stream is reached or *len* bytes are read.

Methods inherited from class java.io.FilterInputStream
`available, close, mark, markSupported, read, reset, skip`

Methods inherited from class java.lang.Object
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

Base64.InputStream

```
public Base64.InputStream(java.io.InputStream in)
```

Constructs a `Base64#InputStream` in DECODE mode.

Parameters:
`in` - the `InputStream` from which to read data.
Since:
1.3

Base64.InputStream

```
public Base64.InputStream(java.io.InputStream in,
                          boolean encode)
```

Constructs a `Base64#InputStream` in either ENCODE or DECODE mode.

Parameters:
`in` - the `InputStream` from which to read data.
`encode` - Conversion direction
Since:
1.3
See Also:
`Base64.ENCODE`, `Base64.DECODE`

Base64.InputStream

```
public Base64.InputStream(java.io.InputStream in,
                          boolean encode,
                          boolean breakLines)
```

Constructs a `Base64#InputStream` in either ENCODE or DECODE mode.

Parameters:

`in` - the `InputStream` from which to read data.
`encode` - Conversion direction
`breakLines` - Break lines at less than 80 characters.

Since:

1.3

See Also:

`Base64.ENCODE`, `Base64.DECODE`

Method Detail

read

```
public int read()
         throws java.io.IOException
```

Reads enough of the input stream to convert to/from Base64 and returns the next byte.

Overrides:

`read` in class `java.io.FilterInputStream`

Returns:

next byte

`java.io.IOException`

Since:

1.3

---

read

```
public int read(byte[] dest,
                int off,
                int len)
         throws java.io.IOException
```

Calls `read()` repeatedly until the end of stream is reached or *len* bytes are read. Returns number of bytes read into array or -1 if end of stream is encountered.

Overrides:

`read` in class `java.io.FilterInputStream`

Parameters:

`dest` - array to hold values
`off` - offset for array
`len` - max number of bytes to read into array

Returns:

bytes read into array or -1 if end of stream is encountered.

`java.io.IOException`

Since:

1.3

Base64.InputStream

Package **Class** Tree Deprecated Index Help
PREV CLASS NEXT CLASS FRAMES NO FRAMES
SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

Class Base64.OutputStream

```
java.lang.Object
  |
  +--java.io.OutputStream
        |
        +--java.io.FilterOutputStream
              |
              +--Base64.OutputStream
```

Enclosing class:
Base64

public static class **Base64.OutputStream**
extends java.io.FilterOutputStream

A Base64#OutputStream will write data to another OutputStream, given in the constructor, and encode/decode to/from Base64 notation on the fly.

Since:
1.3
See Also:
Base64, FilterOutputStream

Field Summary

Fields inherited from class java.io.FilterOutputStream
out

Constructor Summary

Base64.OutputStream(java.io.OutputStream out)	Constructs a Base64#OutputStream in ENCODE mode.
Base64.OutputStream(java.io.OutputStream out, boolean encode)	Constructs a Base64#OutputStream in either ENCODE or DECODE mode.
Base64.OutputStream(java.io.OutputStream out, boolean encode, boolean breakLines)	Constructs a Base64#OutputStream in either ENCODE or DECODE mode.

Method Summary

void	**close**() Flushes and closes (I think, in the superclass) the stream.
void	**flush**() Appropriately pads Base64 notation when encoding or throws an exception if Base64 input is not properly padded when decoding.
void	**write**(byte[] theBytes, int off, int len) Calls write(int) repeatedly until *len* bytes are written.
void	**write**(int theByte) Writes the byte to the output stream after converting to/from Base64 notation.

Methods inherited from class java.io.FilterOutputStream
write

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

Base64.OutputStream

```
public Base64.OutputStream(java.io.OutputStream out)
```

Constructs a Base64#OutputStream in ENCODE mode.

Parameters:
out - the OutputStream to which data will be written.
Since:
1.3

Base64.OutputStream

```
public Base64.OutputStream(java.io.OutputStream out,
                           boolean encode)
```

Constructs a Base64#OutputStream in either ENCODE or DECODE mode.

Parameters:
out - the OutputStream to which data will be written.
encode - Conversion direction
Since:
1.3
See Also:
Base64.ENCODE, Base64.DECODE

Base64.OutputStream

```
public Base64.OutputStream(java.io.OutputStream out,
                           boolean encode,
                           boolean breakLines)
```

Constructs a `Base64#OutputStream` in either ENCODE or DECODE mode.

Parameters:

`out` - the `OutputStream` to which data will be written.
`encode` - Conversion direction
`breakLines` - Break lines to be less than 80 characters.

Since:

1.3

See Also:

`Base64.ENCODE`, `Base64.DECODE`

Method Detail

write

```
public void write(int theByte)
          throws java.io.IOException
```

Writes the byte to the output stream after converting to/from Base64 notation. When encoding, bytes are buffered three at a time before the output stream actually gets a write() call. When decoding, bytes are buffered four at a time.

Overrides:

`write` in class `java.io.FilterOutputStream`

Parameters:

`theByte` - the byte to write
`java.io.IOException`

Since:

1.3

---

write

```
public void write(byte[] theBytes,
                  int off,
                  int len)
           throws java.io.IOException
```

Calls `write(int)` repeatedly until *len* bytes are written.

Overrides:

`write` in class `java.io.FilterOutputStream`

Parameters:

`theBytes` - array from which to read bytes
`off` - offset for array
`len` - max number of bytes to read into array

`java.io.IOException`

Since:

1.3

---

flush

```
public void flush()
          throws java.io.IOException
```

Appropriately pads Base64 notation when encoding or throws an exception if Base64 input is not properly padded when decoding.

Overrides:

`flush` in class `java.io.FilterOutputStream`
`java.io.IOException`

Since:

1.3

---

close

```
public void close()
          throws java.io.IOException
```

Flushes and closes (I think, in the superclass) the stream.

Overrides:

`close` in class `java.io.FilterOutputStream`
`java.io.IOException`

Since:

1.3

Package **Class** Tree Deprecated Index Help

PREV CLASS NEXT CLASS FRAMES NO FRAMES

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

Class BlindSigTest

```
java.lang.Object
  |
  +--java.awt.Component
        |
        +--java.awt.Container
              |
              +--java.awt.Panel
                    |
                    +--java.applet.Applet
                          |
                          +--BlindSigTest
```

All Implemented Interfaces:

javax.accessibility.Accessible, java.awt.event.ActionListener, java.util.EventListener, java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable

---

public class **BlindSigTest**
extends java.applet.Applet
implements java.io.Serializable, java.awt.event.ActionListener

See Also:

Serialized Form

Nested Class Summary

Nested classes inherited from class java.applet.Applet
java.applet.Applet.AccessibleApplet

Nested classes inherited from class java.awt.Panel
java.awt.Panel.AccessibleAWTPanel

Nested classes inherited from class java.awt.Container
java.awt.Container.AccessibleAWTContainer

Nested classes inherited from class java.awt.Component
java.awt.Component.AccessibleAWTComponent, java.awt.Component.BltBufferStrategy, java.awt.Component.FlipBufferStrategy

Field Summary

java.lang.String	helpText

Fields inherited from class java.awt.Component
BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT, TOP_ALIGNMENT

Fields inherited from interface java.awt.image.ImageObserver
ABORT, ALLBITS, ERROR, FRAMEBITS, HEIGHT, PROPERTIES, SOMEBITS, WIDTH

Constructor Summary

BlindSigTest()

Method Summary

void	actionPerformed(java.awt.event.ActionEvent evt)
void	erzeuge()
void	init()

Methods inherited from class java.applet.Applet
destroy, getAccessibleContext, getAppletContext, getAppletInfo, getAudioClip, getAudioClip, getCodeBase, getDocumentBase, getImage, getImage, getLocale, getParameter, getParameterInfo, isActive, newAudioClip, play, play, resize, resize, setStub, showStatus, start, stop

Methods inherited from class java.awt.Panel
addNotify

Methods inherited from class java.awt.Container
add, add, add, add, add, addContainerListener, addImpl, addPropertyChangeListener, addPropertyChangeListener, applyComponentOrientation, areFocusTraversalKeysSet, countComponents, deliverEvent, doLayout, findComponentAt, findComponentAt, getAlignmentX, getAlignmentY, getComponent, getComponentAt, getComponentAt, getComponentCount, getComponents, getContainerListeners, getFocusTraversalKeys, getFocusTraversalPolicy, getInsets, getLayout, getListeners, getMaximumSize, getMinimumSize, getPreferredSize, insets, invalidate, isAncestorOf, isFocusCycleRoot, isFocusCycleRoot, isFocusTraversalPolicySet, layout, list, list, locate, minimumSize, paint, paintComponents, paramString, preferredSize, print, printComponents, processContainerEvent, processEvent, remove, remove, removeAll, removeContainerListener, removeNotify, setFocusCycleRoot, setFocusTraversalKeys, setFocusTraversalPolicy, setFont, setLayout, transferFocusBackward, transferFocusDownCycle, update, validate, validateTree

Methods inherited from class java.awt.Component
action, add, addComponentListener, addFocusListener,

addHierarchyBoundsListener, addHierarchyListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, addMouseWheelListener, bounds, checkImage, checkImage, coalesceEvents, contains, contains, createImage, createImage, createVolatileImage, createVolatileImage, disable, disableEvents, dispatchEvent, enable, enable, enableEvents, enableInputMethods, firePropertyChange, firePropertyChange, firePropertyChange, getBackground, getBounds, getBounds, getColorModel, getComponentListeners, getComponentOrientation, getCursor, getDropTarget, getFocusCycleRootAncestor, getFocusListeners, getFocusTraversalKeysEnabled, getFont, getFontMetrics, getForeground, getGraphics, getGraphicsConfiguration, getHeight, getHierarchyBoundsListeners, getHierarchyListeners, getIgnoreRepaint, getInputContext, getInputMethodListeners, getInputMethodRequests, getKeyListeners, getLocation, getLocation, getLocationOnScreen, getMouseListeners, getMouseMotionListeners, getMouseWheelListeners, getName, getParent, getPeer, getPropertyChangeListeners, getPropertyChangeListeners, getSize, getSize, getToolkit, getTreeLock, getWidth, getX, getY, gotFocus, handleEvent, hasFocus, hide, imageUpdate, inside, isBackgroundSet, isCursorSet, isDisplayable, isDoubleBuffered, isEnabled, isFocusable, isFocusOwner, isFocusTraversable, isFontSet, isForegroundSet, isLightweight, isOpaque, isShowing, isValid, isVisible, keyDown, keyUp, list, list, list, location, lostFocus, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove, mouseUp, move, nextFocus, paintAll, postEvent, prepareImage, prepareImage, printAll, processComponentEvent, processFocusEvent, processHierarchyBoundsEvent, processHierarchyEvent, processInputMethodEvent, processKeyEvent, processMouseEvent, processMouseMotionEvent, processMouseWheelEvent, remove, removeComponentListener, removeFocusListener, removeHierarchyBoundsListener, removeHierarchyListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removeMouseWheelListener, removePropertyChangeListener, removePropertyChangeListener, repaint, repaint, repaint, repaint, requestFocus, requestFocus, requestFocusInWindow, requestFocusInWindow, reshape, setBackground, setBounds, setBounds, setComponentOrientation, setCursor, setDropTarget, setEnabled, setFocusable, setFocusTraversalKeysEnabled, setForeground, setIgnoreRepaint, setLocale, setLocation, setLocation, setName, setSize, setSize, setVisible, show, show, size, toString, transferFocus, transferFocusUpCycle

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

helpText

```
public final java.lang.String helpText
```

See Also:

Constant Field Values

Constructor Detail

BlindSigTest

```
public BlindSigTest()
```

Method Detail

init

```
public void init()
```

Overrides:

`init` in class `java.applet.Applet`

---

actionPerformed

```
public void actionPerformed(java.awt.event.ActionEvent evt)
```

Specified by:

`actionPerformed` in interface `java.awt.event.ActionListener`

---

erzeuge

```
public void erzeuge()
```

Package **Class** Tree Deprecated Index Help
PREV CLASS NEXT CLASS FRAMES NO FRAMES
SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

Class BSigX2Y

```
java.lang.Object
  |
  +--BSigX2Y
```

public class **BSigX2Y**
extends java.lang.Object

This class BSigX2Y is the application for the blind signature on the server. It will be called by signToken.php. It will get the values **x** (blinded token received from the applet), **d** (private key of voter's constituency), **n** (modulus of voter's constituency) from the command line. An optional fourth command line argument can specify the base (radix) of the number system. The default value for the radix is 16. This class will compute and output the value for **y** as `y = x.modPow(d, n);` using the class java.math.BigInteger.

Author: Martin Karl Unger
Last update: 2002-08-27
Compiled using `javac -target 1.1 BSigX2Y.java`

Constructor Summary

`BSigX2Y()` The default constructor only calls super()

Method Summary

static void	`helpmessage()` Is invoked in case of illegal command line arguments and prints a helpmessage to the stream stdout
static void	`main(java.lang.String[] args)` Here the main work is done
static void	`prt(java.lang.String s)` A convenience method for System.out.println(String s)

Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

BSigX2Y

```
public BSigX2Y()
```

The default constructor only calls super()

Method Detail

helpmessage

```
public static void helpmessage()
```

Is invoked in case of illegal command line arguments and prints a helpmessage to the stream stdout

prt

```
public static void prt(java.lang.String s)
```

A convenience method for System.out.println(String s)

main

```
public static void main(java.lang.String[] args)
```

Here the main work is done

Package **Class** Tree Deprecated Index Help
PREV CLASS NEXT CLASS FRAMES NO FRAMES
SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

Class CertLoader

```
java.lang.Object
  |
  +--CertLoader
```

public class **CertLoader**
extends java.lang.Object

Class CertLoader reads a text file from disk which is either a base64-representation of the voter's certificate or his *Personenbindung* (XML file containing his ZMR number). The class StrToCer of this package is used to check whether the read text file contains a certificate.
Author: Martin Karl Unger (2002)

Constructor Summary

`CertLoader()`	This default constructor only calls super()
`CertLoader(int slfCert, int iffCert, java.lang.String certFileName)`	This constructor supplies values for the storage location of the certificate, the input file format of the certificate and the String containig the full path and name of the file which contains the certificate.

Method Summary

java.security.cert.Certificate	`getCert()` Returns the first certificate found in the file called certFileName
java.lang.String	`getCertFileName()` Echoes the argument of SetCertFileName()
int	`getIffCert()` Echoes the argument of setIffCert()
java.lang.String	`getOutputBase64String()` Must contain -----BEGIN CERTIFICATE-----
int	`getSlfCert()` Echoes the argument of setSlfCert()
char[]	`getTextFromDisk()` May contain -----BEGIN CERTIFICATE-----
void	`load()` Does the main work.
void	`setCertFileName(java.lang.String certFileName)` Sets the complete path and name of the file which contains the certificate
void	`setIffCert(int slfCert)` Sets the switch input file format of certificate
void	`setSlfCert(int slfCert)` Sets the switch storage location for certificate

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

CertLoader

```
public CertLoader()
```

This default constructor only calls super()

CertLoader

```
public CertLoader(int slfCert,
                  int iffCert,
                  java.lang.String certFileName)
```

This constructor supplies values for the storage location of the certificate, the input file format of the certificate and the String containig the full path and name of the file which contains the certificate.

Method Detail

setSlfCert

```
public void setSlfCert(int slfCert)
```

Sets the switch storage location for certificate

getSlfCert

```
public int getSlfCert()
```

Echoes the argument of setSlfCert()

setIffCert

```
public void setIffCert(int slfCert)
```

Sets the switch input file format of certificate

---

getIffCert

```
public int getIffCert()
```

Echoes the argument of setIffCert()

---

setCertFileName

```
public void setCertFileName(java.lang.String certFileName)
```

Sets the complete path and name of the file which contains the certificate

---

getCertFileName

```
public java.lang.String getCertFileName()
```

Echoes the argument of SetCertFileName()

---

getCert

```
public java.security.cert.Certificate getCert()
```

Returns the first certificate found in the file called certFileName

---

getTextFromDisk

```
public char[] getTextFromDisk()
```

May contain -----BEGIN CERTIFICATE-----

---

getOutputBase64String

```
public java.lang.String getOutputBase64String()
```

Must contain -----BEGIN CERTIFICATE-----

---

load

```
public void load()
         throws java.io.IOException,
                java.lang.IllegalArgumentException,
                java.security.cert.CertificateException,
                java.lang.Exception
```

Does the main work.

```
java.io.IOException
java.lang.IllegalArgumentException
java.security.cert.CertificateException
java.lang.Exception
```

Package **Class** Tree Deprecated Index Help

PREV CLASS NEXT CLASS FRAMES NO FRAMES

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

Class ElvisCup

```
java.lang.Object
  |
  +--java.awt.Component
        |
        +--java.awt.Container
              |
              +--java.awt.Panel
                    |
                    +--java.applet.Applet
                          |
                          +--ElvisCup
```

All Implemented Interfaces:
javax.accessibility.Accessible, java.awt.event.ActionListener, java.util.EventListener, java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable

public class **ElvisCup**
extends java.applet.Applet
implements java.awt.event.ActionListener

This applet does the token blinding for the blind signature, performs all communication with the server and the SecurityKapsel, generates the electronic voting permit and stores it on disk.

Author: Martin Karl Unger
e-mail: martinkarlunger@yahoo.com
Last Update: 2002-11-05

See Also:
Serialized Form

Nested Class Summary

Nested classes inherited from class java.applet.Applet
java.applet.Applet.AccessibleApplet

Nested classes inherited from class java.awt.Panel
java.awt.Panel.AccessibleAWTPanel

Nested classes inherited from class java.awt.Container

java.awt.Container.AccessibleAWTContainer

Nested classes inherited from class java.awt.Component
java.awt.Component.AccessibleAWTComponent, java.awt.Component.BltBufferStrategy, java.awt.Component.FlipBufferStrategy

Field Summary

Fields inherited from class java.awt.Component
BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT, TOP_ALIGNMENT

Fields inherited from interface java.awt.image.ImageObserver
ABORT, ALLBITS, ERROR, FRAMEBITS, HEIGHT, PROPERTIES, SOMEBITS, WIDTH

Constructor Summary

ElvisCup()

Method Summary

void	actionPerformed(java.awt.event.ActionEvent evt) This method implements the ActionListener so that the applet will react to the commands which the user issues by pressing the buttons.
void	aP_1_loadCertificate() This method will load a text file which either contains a Personenbindung in XML format or the voter´s certificate in base64 format.
void	aP_2_sendCertificate() This method will send the text file with the Personenbindung or voter certificate to the server.
void	aP_3_signTokenTcbss() This method will use the SecurityKapsel for signing the blinded tokenT2 which will be sent to the Trust Center Blind Signature Server.
void	aP_4_sendTokenTcbss() This method will send the blinded tokenT2 to the Trust Center Blind Signature Server.
void	aP_5_signTokenRs() This method will use the SecurityKapsel to sign the blinded tokenT for the Registration Server.
void	aP_6_sendTokenRs() This method will send the blinded tokenT to the Registration Server.
void	aP_7_saveToken() This method will save the electronic voting permission onto the

	hard disk.
void	**aP_99_abbruch**() This method will terminate the applet upon request of the user.
void	**init**()

Methods inherited from class java.applet.Applet

destroy, getAccessibleContext, getAppletContext, getAppletInfo, getAudioClip, getAudioClip, getCodeBase, getDocumentBase, getImage, getImage, getLocale, getParameter, getParameterInfo, isActive, newAudioClip, play, play, resize, resize, setStub, showStatus, start, stop

Methods inherited from class java.awt.Panel

addNotify

Methods inherited from class java.awt.Container

add, add, add, add, add, addContainerListener, addImpl, addPropertyChangeListener, addPropertyChangeListener, applyComponentOrientation, areFocusTraversalKeysSet, countComponents, deliverEvent, doLayout, findComponentAt, findComponentAt, getAlignmentX, getAlignmentY, getComponent, getComponentAt, getComponentAt, getComponentCount, getComponents, getContainerListeners, getFocusTraversalKeys, getFocusTraversalPolicy, getInsets, getLayout, getListeners, getMaximumSize, getMinimumSize, getPreferredSize, insets, invalidate, isAncestorOf, isFocusCycleRoot, isFocusCycleRoot, isFocusTraversalPolicySet, layout, list, list, locate, minimumSize, paint, paintComponents, paramString, preferredSize, print, printComponents, processContainerEvent, processEvent, remove, remove, removeAll, removeContainerListener, removeNotify, setFocusCycleRoot, setFocusTraversalKeys, setFocusTraversalPolicy, setFont, setLayout, transferFocusBackward, transferFocusDownCycle, update, validate, validateTree

Methods inherited from class java.awt.Component

action, add, addComponentListener, addFocusListener, addHierarchyBoundsListener, addHierarchyListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, addMouseWheelListener, bounds, checkImage, checkImage, coalesceEvents, contains, contains, createImage, createImage, createVolatileImage, createVolatileImage, disable, disableEvents, dispatchEvent, enable, enable, enableEvents, enableInputMethods, firePropertyChange, firePropertyChange, firePropertyChange, getBackground, getBounds, getBounds, getColorModel, getComponentListeners, getComponentOrientation, getCursor, getDropTarget, getFocusCycleRootAncestor, getFocusListeners, getFocusTraversalKeysEnabled, getFont, getFontMetrics, getForeground, getGraphics, getGraphicsConfiguration, getHeight, getHierarchyBoundsListeners, getHierarchyListeners, getIgnoreRepaint, getInputContext, getInputMethodListeners, getInputMethodRequests, getKeyListeners, getLocation, getLocation, getLocationOnScreen, getMouseListeners, getMouseMotionListeners, getMouseWheelListeners, getName, getParent, getPeer, getPropertyChangeListeners, getPropertyChangeListeners, getSize, getSize, getToolkit, getTreeLock, getWidth, getX, getY, gotFocus, handleEvent, hasFocus, hide, imageUpdate, inside, isBackgroundSet, isCursorSet, isDisplayable, isDoubleBuffered, isEnabled, isFocusable, isFocusOwner, isFocusTraversable, isFontSet, isForegroundSet, isLightweight, isOpaque, isShowing, isValid, isVisible, keyDown, keyUp, list, list, list, location, lostFocus, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove, mouseUp, move, nextFocus, paintAll, postEvent, prepareImage, prepareImage, printAll, processComponentEvent, processFocusEvent, processHierarchyBoundsEvent, processHierarchyEvent, processInputMethodEvent, processKeyEvent,

processMouseEvent, processMouseMotionEvent, processMouseWheelEvent, remove, removeComponentListener, removeFocusListener, removeHierarchyBoundsListener, removeHierarchyListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removeMouseWheelListener, removePropertyChangeListener, removePropertyChangeListener, repaint, repaint, repaint, repaint, requestFocus, requestFocus, requestFocusInWindow, requestFocusInWindow, reshape, setBackground, setBounds, setBounds, setComponentOrientation, setCursor, setDropTarget, setEnabled, setFocusable, setFocusTraversalKeysEnabled, setForeground, setIgnoreRepaint, setLocale, setLocation, setLocation, setName, setSize, setSize, setVisible, show, show, size, toString, transferFocus, transferFocusUpCycle

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

ElvisCup

public **ElvisCup**()

Method Detail

init

public void **init**()

Overrides:
`init` in class `java.applet.Applet`

---

actionPerformed

public void **actionPerformed**(java.awt.event.ActionEvent evt)

This method implements the ActionListener so that the applet will react to the commands which the user issues by pressing the buttons.

Specified by:
`actionPerformed` in interface `java.awt.event.ActionListener`

---

aP_1_loadCertificate

public void **aP_1_loadCertificate**()

This method will load a text file which either contains a Personenbindung in XML format or the voter´s certificate in base64 format.

---

aP_2_sendCertificate

```
public void aP_2_sendCertificate()
```

This method will send the text file with the Personenbindung or voter certificate to the server.

---

aP_3_signTokenTcbss

```
public void aP_3_signTokenTcbss()
```

This method will use the SecurityKapsel for signing the blinded tokenT2 which will be sent to the Trust Center Blind Signature Server.

---

aP_4_sendTokenTcbss

```
public void aP_4_sendTokenTcbss()
```

This method will send the blinded tokenT2 to the Trust Center Blind Signature Server.

---

aP_5_signTokenRs

```
public void aP_5_signTokenRs()
```

This method will use the SecurityKapsel to sign the blinded tokenT for the Registration Server.

---

aP_6_sendTokenRs

```
public void aP_6_sendTokenRs()
```

This method will send the blinded tokenT to the Registration Server.

---

aP_7_saveToken

```
public void aP_7_saveToken()
```

This method will save the electronic voting permission onto the hard disk.

---

aP_99_abbruch

```
public void aP_99_abbruch()
```


This method will terminate the applet upon request of the user.

Class ElvisCup Handout

public class **ElvisCup**
extends java.applet.Applet
implements java.awt.event.ActionListener

This applet does the token blinding for the blind signature, performs all communication with the server and the SecurityKapsel, generates the electronic voting permit and stores it on disk.

Author: Martin Karl Unger
e-mail: martinkarlunger@yahoo.com
Last Update: 2002-11-05

Method Detail

init

`public void init()`

This method will read the applet parms coming from the html-file which invoked this applet, configure the appearance of this applet according to these parms and it will build the user interface of this applet.

Overrides:
`init` in class `java.applet.Applet`

---

actionPerformed

`public void actionPerformed(java.awt.event.ActionEvent evt)`

This method implements the ActionListener so that the applet will react to the commands which the user issues by pressing the buttons.

Specified by:
`actionPerformed` in interface `java.awt.event.ActionListener`

---

aP_1_loadCertificate

`public void aP_1_loadCertificate()`

This method will load a text file which either contains a Personenbindung in XML format or the voter´s certificate in base64 format.

---

aP_2_sendCertificate

`public void aP_2_sendCertificate()`

This method will send the text file with the Personenbindung or voter certificate to the server.

---

aP_3_signTokenTcbss

`public void aP_3_signTokenTcbss()`

This method will use the SecurityKapsel for signing the blinded tokenT2 which will be sent to the Trust Center Blind Signature Server.

---

aP_4_sendTokenTcbss

`public void aP_4_sendTokenTcbss()`

This method will send the blinded tokenT2 to the Trust Center Blind Signature Server.

---

aP_5_signTokenRs

`public void aP_5_signTokenRs()`

This method will use the SecurityKapsel to sign the blinded tokenT for the Registration Server.

---

aP_6_sendTokenRs

`public void aP_6_sendTokenRs()`

This method will send the blinded tokenT to the Registration Server.

---

aP_7_saveToken

`public void aP_7_saveToken()`

This method will save the electronic voting permission onto the hard disk.

---

aP_99_abbruch

`public void aP_99_abbruch()`

This method will terminate the applet upon request of the user.

Class KeyGen1

```
java.lang.Object
  |
  +--java.awt.Component
        |
        +--java.awt.Container
              |
              +--java.awt.Panel
                    |
                    +--java.applet.Applet
                          |
                          +--KeyGen1
```

All Implemented Interfaces:
javax.accessibility.Accessible, java.awt.event.ActionListener, java.util.EventListener, java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable

public class **KeyGen1**
extends java.applet.Applet
implements java.io.Serializable, java.awt.event.ActionListener

See Also:
Serialized Form

Nested Class Summary

Nested classes inherited from class java.applet.Applet
java.applet.Applet.AccessibleApplet

Nested classes inherited from class java.awt.Panel
java.awt.Panel.AccessibleAWTPanel

Nested classes inherited from class java.awt.Container
java.awt.Container.AccessibleAWTContainer

Nested classes inherited from class java.awt.Component
java.awt.Component.AccessibleAWTComponent, java.awt.Component.BltBufferStrategy, java.awt.Component.FlipBufferStrategy

Field Summary

java.lang.String	**helpText**

Fields inherited from class java.awt.Component
BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT, TOP_ALIGNMENT

Fields inherited from interface java.awt.image.ImageObserver
ABORT, ALLBITS, ERROR, FRAMEBITS, HEIGHT, PROPERTIES, SOMEBITS, WIDTH

Constructor Summary

KeyGen1()

Method Summary

void	**actionPerformed**(java.awt.event.ActionEvent evt)
void	**destroy**()
void	**erzeuge**()
void	**init**()
void	**start**()
void	**stop**()

Methods inherited from class java.applet.Applet
getAccessibleContext, getAppletContext, getAppletInfo, getAudioClip, getAudioClip, getCodeBase, getDocumentBase, getImage, getImage, getLocale, getParameter, getParameterInfo, isActive, newAudioClip, play, play, resize, resize, setStub, showStatus

Methods inherited from class java.awt.Panel
addNotify

Methods inherited from class java.awt.Container
add, add, add, add, add, addContainerListener, addImpl, addPropertyChangeListener, addPropertyChangeListener, applyComponentOrientation, areFocusTraversalKeysSet, countComponents, deliverEvent, doLayout, findComponentAt, findComponentAt, getAlignmentX, getAlignmentY, getComponent, getComponentAt, getComponentAt, getComponentCount, getComponents, getContainerListeners, getFocusTraversalKeys, getFocusTraversalPolicy, getInsets, getLayout, getListeners, getMaximumSize, getMinimumSize, getPreferredSize, insets, invalidate, isAncestorOf, isFocusCycleRoot, isFocusCycleRoot, isFocusTraversalPolicySet, layout, list,

list, locate, minimumSize, paint, paintComponents, paramString, preferredSize, print, printComponents, processContainerEvent, processEvent, remove, remove, removeAll, removeContainerListener, removeNotify, setFocusCycleRoot, setFocusTraversalKeys, setFocusTraversalPolicy, setFont, setLayout, transferFocusBackward, transferFocusDownCycle, update, validate, validateTree

Methods inherited from class java.awt.Component

action, add, addComponentListener, addFocusListener, addHierarchyBoundsListener, addHierarchyListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, addMouseWheelListener, bounds, checkImage, checkImage, coalesceEvents, contains, contains, createImage, createImage, createVolatileImage, createVolatileImage, disable, disableEvents, dispatchEvent, enable, enable, enableEvents, enableInputMethods, firePropertyChange, firePropertyChange, firePropertyChange, getBackground, getBounds, getBounds, getColorModel, getComponentListeners, getComponentOrientation, getCursor, getDropTarget, getFocusCycleRootAncestor, getFocusListeners, getFocusTraversalKeysEnabled, getFont, getFontMetrics, getForeground, getGraphics, getGraphicsConfiguration, getHeight, getHierarchyBoundsListeners, getHierarchyListeners, getIgnoreRepaint, getInputContext, getInputMethodListeners, getInputMethodRequests, getKeyListeners, getLocation, getLocation, getLocationOnScreen, getMouseListeners, getMouseMotionListeners, getMouseWheelListeners, getName, getParent, getPeer, getPropertyChangeListeners, getPropertyChangeListeners, getSize, getSize, getToolkit, getTreeLock, getWidth, getX, getY, gotFocus, handleEvent, hasFocus, hide, imageUpdate, inside, isBackgroundSet, isCursorSet, isDisplayable, isDoubleBuffered, isEnabled, isFocusable, isFocusOwner, isFocusTraversable, isFontSet, isForegroundSet, isLightweight, isOpaque, isShowing, isValid, isVisible, keyDown, keyUp, list, list, list, location, lostFocus, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove, mouseUp, move, nextFocus, paintAll, postEvent, prepareImage, prepareImage, printAll, processComponentEvent, processFocusEvent, processHierarchyBoundsEvent, processHierarchyEvent, processInputMethodEvent, processKeyEvent, processMouseEvent, processMouseMotionEvent, processMouseWheelEvent, remove, removeComponentListener, removeFocusListener, removeHierarchyBoundsListener, removeHierarchyListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removeMouseWheelListener, removePropertyChangeListener, removePropertyChangeListener, repaint, repaint, repaint, repaint, requestFocus, requestFocus, requestFocusInWindow, requestFocusInWindow, reshape, setBackground, setBounds, setBounds, setComponentOrientation, setCursor, setDropTarget, setEnabled, setFocusable, setFocusTraversalKeysEnabled, setForeground, setIgnoreRepaint, setLocale, setLocation, setLocation, setName, setSize, setSize, setVisible, show, show, size, toString, transferFocus, transferFocusUpCycle

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

helpText

public final java.lang.String **helpText**

See Also:

Constant Field Values

Constructor Detail

KeyGen1

public **KeyGen1**()

Method Detail

init

public void **init**()

Overrides:

init in class java.applet.Applet

---

start

public void **start**()

Overrides:

start in class java.applet.Applet

---

stop

public void **stop**()

Overrides:

stop in class java.applet.Applet

---

destroy

public void **destroy**()

Overrides:

destroy in class java.applet.Applet

---

actionPerformed

public void **actionPerformed**(java.awt.event.ActionEvent evt)

Specified by:

actionPerformed in interface java.awt.event.ActionListener

---

erzeuge

```
public void erzeuge()
```

Package **Class** Tree Deprecated Index Help
PREV CLASS NEXT CLASS FRAMES NO FRAMES
SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

Class StrToCer

```
java.lang.Object
  |
  +--StrToCer
```

public class **StrToCer**
extends java.lang.Object

This class converts a String (which contains a base64-encoded Certificate) into an instance of the class Certificate.
Author: Martin Karl Unger
Last update: 2002-09-15

Constructor Summary

`StrToCer()`	The default constructor specifies the type of the certificate as "X.509"
`StrToCer(java.lang.String inputString)`	Calls the default constructor and sets the String which contains the certificate
`StrToCer(java.lang.String inputString, java.lang.String inputCertType)`	Sets the String with the base64-representation of the certificate and the type of the certificate

Method Summary

java.security.cert.Certificate	`convert()` Here the main work is done.
static java.security.cert.Certificate	`convert(java.lang.String inputString)` A convenience method for convert().
static java.security.cert.Certificate	`convert(java.lang.String inputString, java.lang.String inputCertType)` Another convenience method for convert ().
java.lang.String	`getInputCertType()` Echoes the argument of setInputCertType ( )
java.lang.String	`getInputString()` Echoes the argument of setInputString( )
java.security.cert.Certificate	`getOutputCert()` Returns the certificate which was generated
java.lang.String	`getOutputString()` Returns a base64-representation of the certificate enclosed between -----BEGIN CERTIFICATE----- and -----END CERTIFICATE-----
void	`setInputCertType(java.lang.String inputCertType)` Sets the type of the certificate
void	`setInputString(java.lang.String inputString)` Sets the value for the String which contains the base64-representation of the certificate and may contain the texts -----BEGIN CERTIFICATE----- and -----END CERTIFICATE-----

Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

StrToCer

```
public StrToCer()
```

The default constructor specifies the type of the certificate as "X.509"

StrToCer

```
public StrToCer(java.lang.String inputString)
```

Calls the default constructor and sets the String which contains the certificate

StrToCer

```
public StrToCer(java.lang.String inputString,
                java.lang.String inputCertType)
```

Sets the String with the base64-representation of the certificate and the type of the certificate

Method Detail

getInputString

```
public java.lang.String getInputString()
```

Echoes the argument of setInputString( )

---

setInputString

```
public void setInputString(java.lang.String inputString)
```

Sets the value for the String which contains the base64-representation of the certificate and may contain the texts -----BEGIN CERTIFICATE----- and -----END CERTIFICATE-----

---

getOutputCert

```
public java.security.cert.Certificate getOutputCert()
```

Returns the certificate which was generated

---

getInputCertType

```
public java.lang.String getInputCertType()
```

Echoes the argument of setInputCertType( )

---

setInputCertType

```
public void setInputCertType(java.lang.String inputCertType)
```

Sets the type of the certificate

---

getOutputString

```
public java.lang.String getOutputString()
```

Returns a base64-representation of the certificate enclosed between ----- BEGIN CERTIFICATE----- and -----END CERTIFICATE-----

---

convert

```
public static java.security.cert.Certificate convert(java.lang.String inputStrin
                                                     throws java.lang.IllegalArgumentEx
                                                            java.security.cert.Certific
```


A convenience method for convert().

```
java.lang.IllegalArgumentException
java.security.cert.CertificateException
```

---

convert

```
public static java.security.cert.Certificate convert(java.lang.String inputStrin
                                                     java.lang.String inputCertTy
                                                     throws java.lang.IllegalArgumentEx
                                                            java.security.cert.Certific
```

Another convenience method for convert().

```
java.lang.IllegalArgumentException
java.security.cert.CertificateException
```

---

convert

```
public java.security.cert.Certificate convert()
                                      throws java.lang.IllegalArgumentException,
                                             java.security.cert.CertificateExcep
```

Here the main work is done.

```
java.lang.IllegalArgumentException
java.security.cert.CertificateException
```

Package **Class** Tree Deprecated Index Help

PREV CLASS NEXT CLASS FRAMES NO FRAMES

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

Class StrToCerTest

```
java.lang.Object
  |
  +--java.awt.Component
        |
        +--java.awt.Container
              |
              +--java.awt.Panel
                    |
                    +--java.applet.Applet
                          |
                          +--StrToCerTest
```

All Implemented Interfaces:

javax.accessibility.Accessible, java.awt.event.ActionListener, java.util.EventListener, java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable

---

public class **StrToCerTest**
extends java.applet.Applet
implements java.io.Serializable, java.awt.event.ActionListener

This applet tests my class StrToCer.
Author: Martin Karl Unger
Last update: 2002-09-15

See Also:

Serialized Form

Nested Class Summary

Nested classes inherited from class java.applet.Applet
java.applet.Applet.AccessibleApplet

Nested classes inherited from class java.awt.Panel
java.awt.Panel.AccessibleAWTPanel

Nested classes inherited from class java.awt.Container
java.awt.Container.AccessibleAWTContainer

Nested classes inherited from class java.awt.Component

java.awt.Component.AccessibleAWTComponent, java.awt.Component.BltBufferStrategy, java.awt.Component.FlipBufferStrategy

Field Summary

java.lang.String	**helpText** This text will be displayed in the TextArea when the user presses the button called *Help*
java.lang.String	**testData** This base64-representation of an X.509-certificate starts with -----BEGIN CERTIFICATE----- and will be displayed in the TextArea when the user presses the button called *testdata*
java.lang.String	**testData2** This part of a signed XML file contains the XML tag <dsig:X509Certificate> and will be shown in the TextArea when the user presses the button called *testdata 2*

Fields inherited from class java.awt.Component
BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT, TOP_ALIGNMENT

Fields inherited from interface java.awt.image.ImageObserver
ABORT, ALLBITS, ERROR, FRAMEBITS, HEIGHT, PROPERTIES, SOMEBITS, WIDTH

Constructor Summary

StrToCerTest() This standard constructor only calls super()

Method Summary

void	**actionPerformed**(java.awt.event.ActionEvent evt) Implements the ActionListener interface
void	**actionPStart**(java.awt.event.ActionEvent evt) Takes the string which is displayed in the TextArea and converts it into an X.509-Certificate using my class StrToCer
void	**init**() Initializes this applet

Methods inherited from class java.applet.Applet
destroy, getAccessibleContext, getAppletContext, getAppletInfo, getAudioClip, getAudioClip, getCodeBase, getDocumentBase, getImage, getImage, getLocale, getParameter, getParameterInfo, isActive, newAudioClip, play, play, resize, resize, setStub, showStatus, start, stop

Methods inherited from class java.awt.Panel
addNotify

Methods inherited from class java.awt.Container
add, add, add, add, add, addContainerListener, addImpl, addPropertyChangeListener, addPropertyChangeListener, applyComponentOrientation, areFocusTraversalKeysSet, countComponents, deliverEvent, doLayout, findComponentAt, findComponentAt, getAlignmentX, getAlignmentY, getComponent, getComponentAt, getComponentAt, getComponentCount, getComponents, getContainerListeners, getFocusTraversalKeys, getFocusTraversalPolicy, getInsets, getLayout, getListeners, getMaximumSize, getMinimumSize, getPreferredSize, insets, invalidate, isAncestorOf, isFocusCycleRoot, isFocusCycleRoot, isFocusTraversalPolicySet, layout, list, list, locate, minimumSize, paint, paintComponents, paramString, preferredSize, print, printComponents, processContainerEvent, processEvent, remove, remove, removeAll, removeContainerListener, removeNotify, setFocusCycleRoot, setFocusTraversalKeys, setFocusTraversalPolicy, setFont, setLayout, transferFocusBackward, transferFocusDownCycle, update, validate, validateTree

Methods inherited from class java.awt.Component
action, add, addComponentListener, addFocusListener, addHierarchyBoundsListener, addHierarchyListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, addMouseWheelListener, bounds, checkImage, checkImage, coalesceEvents, contains, contains, createImage, createImage, createVolatileImage, createVolatileImage, disable, disableEvents, dispatchEvent, enable, enable, enableEvents, enableInputMethods, firePropertyChange, firePropertyChange, firePropertyChange, getBackground, getBounds, getBounds, getColorModel, getComponentListeners, getComponentOrientation, getCursor, getDropTarget, getFocusCycleRootAncestor, getFocusListeners, getFocusTraversalKeysEnabled, getFont, getFontMetrics, getForeground, getGraphics, getGraphicsConfiguration, getHeight, getHierarchyBoundsListeners, getHierarchyListeners, getIgnoreRepaint, getInputContext, getInputMethodListeners, getInputMethodRequests, getKeyListeners, getLocation, getLocation, getLocationOnScreen, getMouseListeners, getMouseMotionListeners, getMouseWheelListeners, getName, getParent, getPeer, getPropertyChangeListeners, getPropertyChangeListeners, getSize, getSize, getToolkit, getTreeLock, getWidth, getX, getY, gotFocus, handleEvent, hasFocus, hide, imageUpdate, inside, isBackgroundSet, isCursorSet, isDisplayable, isDoubleBuffered, isEnabled, isFocusable, isFocusOwner, isFocusTraversable, isFontSet, isForegroundSet, isLightweight, isOpaque, isShowing, isValid, isVisible, keyDown, keyUp, list, list, list, location, lostFocus, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove, mouseUp, move, nextFocus, paintAll, postEvent, prepareImage, prepareImage, printAll, processComponentEvent, processFocusEvent, processHierarchyBoundsEvent, processHierarchyEvent, processInputMethodEvent, processKeyEvent, processMouseEvent, processMouseMotionEvent, processMouseWheelEvent, remove, removeComponentListener, removeFocusListener, removeHierarchyBoundsListener, removeHierarchyListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removeMouseWheelListener, removePropertyChangeListener, removePropertyChangeListener, repaint, repaint, repaint, repaint, requestFocus, requestFocus, requestFocusInWindow, requestFocusInWindow, reshape, setBackground, setBounds, setBounds, setComponentOrientation, setCursor, setDropTarget, setEnabled, setFocusable, setFocusTraversalKeysEnabled, setForeground, setIgnoreRepaint, setLocale, setLocation, setLocation, setName, setSize, setSize, setVisible, show, show, size, toString, transferFocus, transferFocusUpCycle

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

helpText

```
public java.lang.String helpText
```

This text will be displayed in the TextArea when the user presses the button called *Help*

---

testData

```
public java.lang.String testData
```

This base64-representation of an X.509-certificate starts with -----BEGIN CERTIFICATE----- and will be displayed in the TextArea when the user presses the button called *testdata*

---

testData2

```
public java.lang.String testData2
```

This part of a signed XML file contains the XML tag <dsig:X509Certificate> and will be shown in the TextArea when the user presses the button called *testdata 2*

Constructor Detail

StrToCerTest

```
public StrToCerTest()
```

This standard constructor only calls super()

Method Detail

init

```
public void init()
```

Initializes this applet

Overrides:

`init` in class `java.applet.Applet`

---

actionPerformed

```
public void actionPerformed(java.awt.event.ActionEvent evt)
```

Implements the ActionListener interface

Specified by:

`actionPerformed` in interface `java.awt.event.ActionListener`

---

actionPStart

```
public void actionPStart(java.awt.event.ActionEvent evt)
```

Takes the string which is displayed in the TextArea and converts it into an X.509-Certificate using my class StrToCer

---

Package **Class** Tree Deprecated Index Help

PREV CLASS NEXT CLASS FRAMES NO FRAMES

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

Class XmlBase64

```
java.lang.Object
  |
  +--XmlBase64
```

public class **XmlBase64**
extends java.lang.Object

This class converts a String into an XML-file with the String coded in base64-format using the class Base64 which is supplied here. The result will be sent to the SecurityKapsel in order to be signed. See the documentation of the **Security Layer** for details.
Author: Martin Karl Unger (2002)

Field Summary

java.lang.String	**xmlBody** Second part of XML file to sign with base64 content.
java.lang.String	**xmlFooter** Third part of XML file to sign.
java.lang.String	**xmlHeader** First part of XML file to sign.

Constructor Summary

XmlBase64() This default constructor only calls super()
XmlBase64(java.lang.String toSign) Supplies the String which will be signed

Method Summary

void	**convert**() Does the main work using the class Base64
static java.lang.String	**convert**(java.lang.String toSign) Instantiates this class with the String to sign, convert() and returns the signed XML file
java.lang.String	**getInputString**() Echoes the argument of setInputString()
java.lang.String	**getOutputString**() Returns the output of the SecurityKapsel as signed XML file
static void	**main**(java.lang.String[] args) For testing with a command line argument
void	**setInputString**(java.lang.String toSign) Supplies the String which will be signed

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

xmlHeader

public java.lang.String **xmlHeader**

First part of XML file to sign.

xmlBody

public java.lang.String **xmlBody**

Second part of XML file to sign with base64 content.

xmlFooter

public java.lang.String **xmlFooter**

Third part of XML file to sign.

Constructor Detail

XmlBase64

public **XmlBase64**()

This default constructor only calls super()

XmlBase64

public **XmlBase64**(java.lang.String toSign)

Supplies the String which will be signed

Method Detail

setInputString

```
public void setInputString(java.lang.String toSign)
```

Supplies the String which will be signed

---

getInputString

```
public java.lang.String getInputString()
```

Echoes the argument of setInputString()

---

getOutputString

```
public java.lang.String getOutputString()
```

Returns the output of the SecurityKapsel as signed XML file

---

convert

```
public static java.lang.String convert(java.lang.String toSign)
```

Instantiates this class with the String to sign, convert() and returns the signed XML file

---

main

```
public static void main(java.lang.String[] args)
```

For testing with a command line argument

---

convert

```
public void convert()
```

Does the main work using the class Base64

XmlBase64

Alle Informationen auch erhältlich unter

www.e-Voting.at