Electronic elections employing DES smartcards

by

Herman Robers

December 1998

IBM Student Chipcard Innovation Team
Under supervision of
Pieter G. Maclaine Pont – IBM Nederland N.V.
Prof.ir. Jaap W. van Till – Delft University of Technology

Abstract

This paper covers the design of a voting protocol which can be used to perform local electronic elections with the use of currently commercial available devices.

In contradiction with other proposed voting schemes the new proposed protocol does not rely on properties of asymmetric cryptographic algorithms like RSA. If needed an asymetric protocol can be used to attain the needed functions in the new protocol. It uses some of the techniques proposed in [Rob98] to authenticate messages without the need of cryptographic keys on public systems.

Design characteristics are anonymously, democratically, non-coercion and public verifiably. Meeting all of these requirements is probably impossible. In any proposed scheme implementing all but one of these requirements is achieved.

A nice implementation feature of the designed system is that the needed technology is already available and widely spread implemented in electronic purse smartcards.

Preface

Like many institutes, Delft University of Technology has some democracy in the government of the organization. Every year a students council is chosen by an election in which all the students can place a vote for a person who may represent him to the university-board. Every two years all employees have to vote for the works council.

Elections are very expensive: Every voter needs to receive a personal invitation by mail (postage and printing costs), several people are needed to run the voting office, the voting offices need to be equipped, lots of security processes, and so on.

At Delft University of Technology all students and staff members received a smartcard called "Campuscard". This card is a version of the *Studenten Chipkaart*, a smartcard issued by the foundation *Stichting Studenten Chipkaart*. Given the fact that all students and employees have been given such a smartcard creates some nice opportunities. The card can be used to pay small amounts at the university like copier, restaurant, candy machines. The card has functions prepared for access control to buildings, rooms, computers and networks. And the students can use the smartcard for remote authentication to the IBG[1]. All functions are optional, educational institutes who have introduced this smartcard may implement only those nessecary, but can also add their own applications. A nice new application in that category would be: electronic elections. Mailing of personal polling cards is no longer needed, elections may even take place at public terminals or the personal computer of the student at home. This reduces the costs of a voting dramatically.

The most obvious problem is that the Campuscard primary has an identifying function; all implemented techniques are used to identify a person. Elections on the other hand have the requirement to be anonymous. At first glance these functions conflict with the election requirements. By application of the techniques described in [Rob98] we are able to solve these problems.

[1] Informatie Beheer Groep, Dutch governmental institution responsible for the administration of scholarships

Contents

Chapter 1

Design of a Voting protocol

Election or voting is a democratic process to give people the possibility to state their opinion about any subject. In most cases it is used to choose the people who represent the mass. But it can also be used to poll the opinion about an important case. Since votings are usually organized by a party who depends on the results, votings have some very special characteristics: it should be anonymous, but at the same time it should be fully auditable. According to [Sch96] the ideal voting protocol has the following requirements:

1. Only authorized voters can vote.
2. No one can vote more than once.
3. No one can determine for whom anyone has voted.
4. No one can duplicate anyone else's vote.
5. No one can change anyone else's vote without being discovered.
6. Every voter can make sure that his vote has been taken into account in the final tabulation.
7. And in some cases: Everyone knows who voted and who did not.

Other publications[Cra96] group the requirements by the following characteristics:

- Accuracy: votes can't be altered (5 above), validated votes can not be eliminated from the final tally (6 above) and it is not possible that an invalid vote is counted in the final tally (1 and 2 above).
- Democracy: Only authorized voters can vote (1 above) and no one can vote more than once (2 above).
- Privacy: it is not possible to determine for whom anyone has voted (3 above) and no voter can prove that he or she voted in a particular way (non-coercion, not fully covered above).
- Verifiability: An external auditing party can verify if the votes have been counted correctly and a voter can determine if his vote was counted correctly (6 above).

Design of a voting protocol that meets all these characteristics is very complex and maybe even impossible. The traditional voting protocol lacks some of these requirements more or less depending on the procedures.

1.1 Voting terminology

In the field of elections a lot of technical terms are used. Before continuing this chapter the used election-terms will be explained.

Voting: The democratic process in which a large population states its opinion about some subject (poll) or person (election).

Election: A voting in which one or more candidates are chosen to represent the voters. All voters may select their favorite candidate and the candidate(s) with the most votes are selected.

Poll: A voting in which an opinion is examined. The voters may choose between yes and no, or may select one of several alternatives.

Vote: Opinion or choice for a person, written on an anonymous ballot. It must not be possible to reveal someone's vote without cooperation of the voter himself.

Voter: The person who casts his vote.

Entitled voter: A person who is allowed to vote. In most cases he may vote or renounce his right to vote.

Election Notification: The invitation an entitled voter receives with which he can authenticate himself at the polling station and may submit his vote.

Ballot: The piece of paper (or an equivalent) on which the voter may select his vote. A ballot should be anonymous: the same for all voters before the selection is written on it, unmarked and unnumbered.

Polling station: The location or building at which the voter is able to vote. The polling station and the procedures at the polling station are inspected by the polling committee.

Polling booth: The separated room in which the voter can fill out his ballot without officials or other people watching what the vote is.

Ballot box: The box in which all ballots are collected. Before the voting it is emptied and sealed with a lead seal. After the voting the polling committee ensures that the ballot box is still sealed. Because the votes of all the voters at the polling station are in the same box the ballots can be considered anonymous.

Polling committee: The officials who are selected to inspect the voting by the voting organizer. The polling committee is composed in such a way that all members inspect each other and are from different political parties. This means that if one of the members tries to tamper with the votes that attempt will be noticed by the other members. To effectively fraud the election results the full polling committee needs to collude.

Tally: The tally determines the results of the voting. It receives the voted ballots and determines how many votes each option has received. The results are published by the tally.

Turnout: The percentage of entitled voters that show up and submit their vote.

1.2 Traditional voting

The traditional voting scheme as shown in figure 1.1 exists of the following phases:
Before the election can take place:

- The election organizing committee makes a selection of voters.

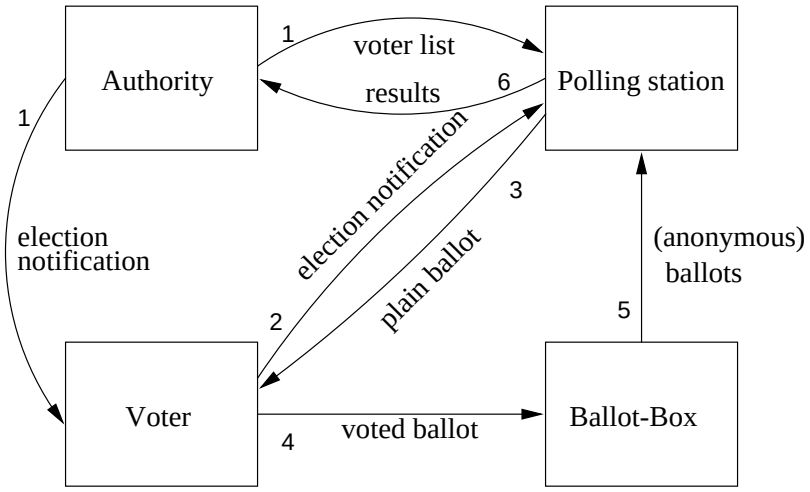

Figure 1.1: Graphic representation of the information flows in the traditional voting process

- The organizing committee has election notifications created and has mailed to the entitled voters.
- Polling stations need to be set up at different locations where voters can submit their vote.

During the election:

- The polling committee verifies the name on the polling card to a list of entitled voters and marks the vote as used. The voter receives an unmarked ballot and will be able to cast his vote anonymously.
- The voter marks his vote on the ballot
- The voter disposes his ballot in the sealed ballot box

After the polling stations are closed:

- The polling committee breaks the seal on the ballot box
- The committee members count the total number of ballots and compare that to the marked number of votes on the voter list.
- The members sort the ballots on submitted vote, and count the votes. Results are submitted to a regional or central tally and added to the final tally results.
- The central tally publishes the results.

A voter will not be able to find out if his vote is taken into account at the final tally which is in contradiction to the requirements of a voting. The other conditions depend strongly on the integrity of the polling committee. The polling committee could cast votes for people who didn't show up, they might mark the ballots and trace back votes to certain people, miscount the votes and so on. In the traditional voting system the polling committee is trusted. To ensure integrity several committee members are needed to perform each task in the polling system. Those are chosen from different political backgrounds to create contradicting interests. Another weak point of the traditional system is the fact that polling cards are sent by mail. Obtaining those polling cards is not that hard when they are in a unlocked mailbox. Possession of the election notification is all you need to cast a vote. The security of this system is based on the notion that people will complain if they didn't receive a notification.

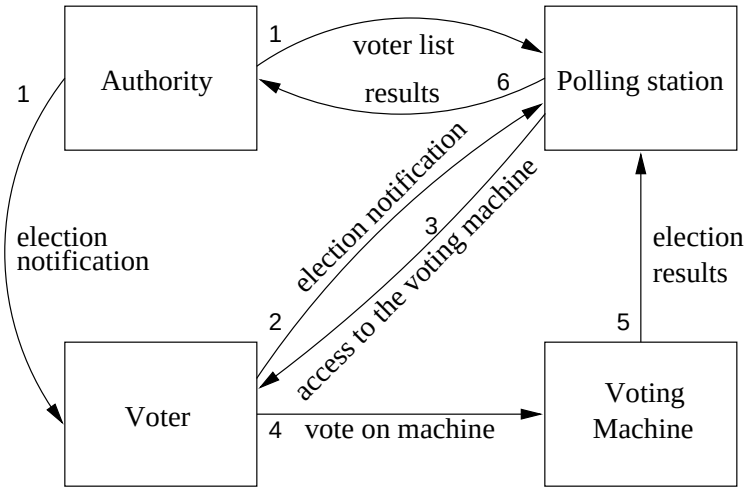

Figure 1.2: Graphic representation of the information flows in the voting process when voting machines are used

1.2.1 Voting with electronic voting machines

Recently electronic voting machines were introduced at Dutch elections. This has resulted in some changes in the phases of the traditional election:

Before the election can take place (not changed):

- The election organizing committee makes a selection of voters.
- The organizing committee has election notifications created and mailed to the voters.
- Setting up polling stations at different locations where voters can submit their vote.

Until now it the procedures are still the same. During the election:

- The polling committee verifies the name on the polling card to a list of entitled voters and marks the vote as used. The voter receives a receipt with which he can cast a vote at the voting machine
- The voter delivers his receipt at the voting machine operator and enters the private voting booth.
- The operator unlocks the machine
- The voter presses the button of his choice, his choice appears on the display of the machine. The operators display shows that a choice was made.
- If the vote is correct the voter has to press the red vote-button to confirm his vote. The operators display shows: voted.
- The vote is stored in a tamper-proof module (about the size of a package of cigarettes) in the voting machine. The voting machine is locked and placed back in the initial state.

After the polling stations are closed:

- The polling committee has the voting machine print the voting results.
- The committee compares the total recorded votes to the marked number of votes on the voter list.

- The committee submits the printout and the tamper-proof module with recorded votes to a regional or central tally where it is added to the final tally results.
- The central tally publishes the results.

The system is illustrated in figure 1.2. In this system the possibility of miscounting votes is eliminated. Because the votes are not recorded with a timestamp or sequence-number, backtracking of votes to individuals by the polling committee is no longer possible. Stealing of election notifications and casting of unused votes is still possible if the committee cheats together.

A new to be designed voting system should not suffer more of these weaknesses, and preferable solve some.

1.3 Known secure electronic voting systems

Several voting schemes have been proposed, some have been implemented as well. Most voting schemes are unable to satisfy all design characteristics.

1.3.1 Sensus

Lorrie Faith Cranor describes in [Cra96] an implementation of a voting scheme proposed by Fujioka, Okamoto, and Ohta [FOO92]. The scheme uses blind signatures, a method to maintain both security and anonymity. Blind signatures are introduced by Chaum [Cha83] and allow someone to sign a document without knowledge of its contents. This algorithm is mostly visualized by an envelope with carbon paper inside. Somebody else can place a signature on the envelope and through the carbon copy on the document at the same time. If the envelope is still sealed, you can verify the person signing the document could not have taken notice of what is in the document. If you remove the document from the envelope the signature remains attached to the document.

In the Sensus protocol a voter composes a ballot and encrypts it with a chosen key. That encrypted ballot is blinded with a chosen blindening factor. He signs the blinded, encrypted ballot with his secret key and submits it to the voting authority. The voting authority verifies the signature with the voter's public key and verifies the identity against the list of valid voters. If the voter is allowed to vote and has not already casted his vote the encrypted ballot is signed by the voting authority, marking it as a valid vote. Because the ballot is encrypted with a key not known to the authority, the latter is unable to determine which vote is in the ballot. Because of the blindening factor the authority can't even reveal how the encrypted ballot looks. This is important because the decryption key is published later on. The voter is given back his ballot and removes the blindening layer. What remains is a ballot signed by the voting authority and encrypted with a key chosen by the voter. This encrypted ballot is casted to the tally which verifies the signature with the public key of the authority and signs the ballot as received with its own public key and assigns a receipt number to the ballot. The signed encrypted ballot is returned to the voter who verifies the signature of the tally and publishes in a separate session the decryption-key for the ballot accompanied with the receipt number.

This protocol uses blind signatures which requires some special properties from the used cryptographic algorithms. The blindening process is a multiplication before signing and a division at the end, in algorithms like RSA and ElGamal those operations cancel each other out, in the smartcards DES algorithm they don't. This means that the smartcard cannot improve the voting process by using blind signatures.

1.3.2 Secure, Optimally Efficient Multi-Authority Election Scheme

In [CFSY96] a voting scheme is proposed that uses multi-party computations realize voting requirements. The voter posts an encrypted message accompanied by a compact proof that the message contains a valid vote. Using the proof anyone can verify if the encrypted vote is valid, but is not able to determine what the vote actually is. Decryption can be done with a private

key that is distributed over a number of authorities. This means that none of the authorities can decrypt the message on its own. The authorities must work together to decrypt the encrypted ballots. A disadvantage of this scheme is that there are only two voting options: 1 or -1, which can be considered the representation for "yes" or "no". To offer choice between more candidates, every candidate can be voted "yes" or "no", where only one candidate may receive a "yes"-vote. The received encrypted ballots are multiplied with each other and can be decrypted in one-time. The result of the decryption is the difference between the number of "yes"-votes and the number of "no"-votes. This property in which the decryption of a multiplication of encrypted messages results in the sum of the used plaintexts is called homomorphic encryption. The ElGamal crypto-system based on discrete logarithms satisfies this property. A nice feature is the threshold function which means that a number less than the total of authorities are able to decrypt the ballots together. In this way for example any combination of 10 out of the 15 available authorities are sufficient to decrypt the ballots. An improved version which needs less communication is given in [RRB97].

1.4 Secure electronic voting with the use of DES-smartcards

The reason that we would like to perform elections with DES-smartcards is that this type of card is very widely spread. Fancy new voting schemes employing hot new cryptographic algorithms will need to issue new smartcards to all of its voters. Because issuing those kind of cards is very expensive this is not very attractive. Using space on someone else's smartcard (hitchhiking) is a better solution. Hitchhiking is possible on well designed multi-function smartcards. It provides the possibility to divide the smartcard into multiple parts without the need of a fully trusted party with knowledge of all the data and keys on the card. This creates the opportunity of carrying multiple trusted applications on one card like electronic purse and social security functions without the disadvantage that your bank is able to watch your social security information or that the social security agency can touch your banking information. How this can be achieved is described in [IBM96] and elaborated for the SCK case in [vdL97]. It is even possible to store new or updated keys in the smartcard over an insecure network like the internet.

To design an election scheme we first need to identify the parties involved. First an election organizing party is needed to determine who may vote and about what. This can be a government or a university. The voter and the organizing party have to know each other. In the government case the voters receive a polling card, and identify themselves with an ID-card from the same government. In the university case the students have received a college card which they can use for identification. A third party is the polling station, it is trusted by all other parties and should be organized in such a way that fraud is very difficult.

The election notification is essential in traditional voting to meet several of the requirements for elections. The card gives the polling station the ability to verify if the voter is allowed to vote and by withdrawing the card the possibility to vote twice is eliminated. To detect false polling cards a list of eligible voters is used for a double accounting system. Cards for a chosen identity can't be used because those identities do not appear on the list and copying of cards fails because the identity is marked on the list as used.

Functions in a voting scheme:

- Voting authority: Organizer of the election, determines who may vote and what the voters can vote.
- Lists of entitled voters: Who may vote and at which polling station.
- Polling stations: The physical location at which a voter may cast his vote. This may be a controlled and audited system, more preferably this function should be implementable at any 'insecure' system (i.e. at the student's own PC).
- Voters: The person who is allowed to vote. A voter may vote only once but can also decide not to vote at all.

- Polling booth: An 'anonymous channel', because the ballots of all voters are collected together and 'randomized' when falling out of the polling booth.
- Talliers: Persons who count the votes and calculate the (sub)-tally.

1.5 A new electronic voting scheme with the use of smartcards

The new voting protocol will use three separate entities: the voter, the authority and an anonymizer.

Voter: a piece of software with which an entitled voter can submit a vote. Because software can be easily replaced or adapted to fool the system the most critical operations are delegated to the trusted smartcard. The voter function can be implemented everywhere and must be trusted by the voter person.

Authority: a combination of software and hardware which makes a voting possible. An election is initiated by the authority. The authority has a relationship with all the entitled voters by having a shared key. The shared key is protected by a hardware cryptographic facility and can only be used to write a key into some designated field on the voters smartcard so that the voter can use that written key on data in the smartcard. A relation to the anonymizer consists of a shared key called K_{anon} that can be used for encryption only at the authority.

Anonymizer: separation of the voter and authority. The function of the anonymizer is to publish submitted information. In fact none of the transported messages contains identifying information, but the message in combination with information about the source of the message may reveal additional information. The anonymizer shares a key K_{voter} with the authority with which it can decrypt only (function separation). The messages are published without additional information like order, time and source. The published information may be available to anyone who likes to know and is allowed to view the voting results.

1.5.1 Voting procedure

To submit a vote a voter should perform the following steps:

- Register to take part in the election. The voter receives the information needed to submit the vote such as keys and candidates to choose.
- With this information the voter can calculate his unique and anonymous VOTER_ID using his smartcard.
- The voter selects the candidate of his choice and writes the corresponding CANDIDATE_ID in his smartcard.
- The smartcard generates an authentication code over the CANDIDATE_ID which can be used in combination with the VOTER_ID as ballot.
- The ballot is submitted to the anonymizer and acknowledged by the anonymizer.
- After the election has closed the anonymizer publishes all the information he received. Anyone can calculate the final results from that information.

Note that these steps need not to be performed in one session. To provide more privacy this is even discouraged.

1.5.2 Elaboration of the procedure

Before the election can take place the voting authority has to do the following:

- A list of valid voters should be composed. Voter specific information is supposed to be available from a database at the authority or another party. The total number of entitled voters needs to be published.
- An identifier for the election must be created and published. This might even be a text string describing which election is held. This is called the ELECTION_ID.
- A unique CANDIDATE_ID must be generated for each of the valid choices or candidates. These also have to be published.

The authority or another party has to do the following using cryptographic hardware:

- Generate an unique key K_{voter} for each voter and distribute it to the voters smartcard. Based on the property of modern multi-function smartcards, the key can be safely loaded after issuing the card without the need of a secure channel [IBM96]. This means that the keys can be loaded over the internet or at a public terminal in an entrance hall. The SCK has this ability implemented with the so called LOAD_KEY–command [vdL97].
- Generate a ballot-collection for each voter. The ballot collection is constructed like this:

$$\mathbf{E}_{\mathbf{K_{anon}}} \begin{pmatrix} \text{VOTER\_ID} = MDC(MAC_{K_{voter}}(\text{ELECTION\_ID})) \\ MDC(MAC_{K_{voter}}(\text{CANDIDATE\_ID1})) \\ MDC(MAC_{K_{voter}}(\text{CANDIDATE\_ID2})) \\ MDC(MAC_{K_{voter}}(\text{CANDIDATE\_ID3})) \\ \vdots \\ MDC(MAC_{K_{voter}}(\text{CANDIDATE\_IDN})) \end{pmatrix}$$

The MDC (Modification Detection Code) is used as a public one-way function, this means the MDC value is easy to derive from a known $MAC_{K_{voter}}$ but given an MDC there is no possibility to reveal the MAC it was calculated from. Because the only place where the MAC can be calculated is at a place where K_{voter} is known, we can be sure that if that MAC is published it originates from one of those places. The crypto hardware at the authority is programmed in such a way that only the cascaded operation $MDC(MAC_{K_{voter}}(\ldots))$ can be performed. The key K_{voter} is generated in the cryptographic hardware and stored in the smartcard of the voter. The cryptographic hardware should be limited in such a way that the use of K_{voter} to perform the same operation as the smartcard does is not possible. Limiting cryptographic hardware is possible using control vectors (see appendix A.4). The choice for the MDC function is based on the fact that MDC is a standard function in the IBM product-line of cryptographic hardware, but technically any trusted one-way function can be used.

It is obvious that the VOTER_ID is not retraceable to the corresponding voter because a MAC is performed with K_{voter} and the voter has his smartcard to calculate his own VOTER_ID. At the authority the only possibility is to view which VOTER_ID appears in the ballot-collection when it is generated. This problem is blocked by having the ballot-collection encrypted with K_{anon} before it leaves the crypto-hardware. The key K_{anon} at the authority can only be used to encrypt ballot-collections. Decryption of those ballot-collections is allowed at the anonymizer function, further called anonymizer. If the voter list is sorted, i.e. alphabetically, the authority needs to shuffle the encrypted ballot-collections before sending them to the anonymizer to prevent guessing VOTER_ID based on the sequence.

Before the voting the anonymizer has the task of:

- Decrypting the ballot-collections for all voters with K_{anon}
- Sorting all the ballot-collections on the VOTER_ID

- Publishing a list of all ballot-collections sorted in VOTER_ID

During the election:

- Receiving the voted ballots from the voters
- Acknowledging the reception of a vote by writing some value in the voters smartcard.

After the election:

- Publishing the submitted ballots

The sorting is done for two reasons: a voter can easily find his own ballot-collection and the voting authority can no longer link the VOTER_ID to the voter based on sequence.

To avoid the possibility of the anonymizer to insert fake voters, this function should be partly performed in trusted auditable hardware which can only use K_{anon} to decrypt the ballot-collections. The anonymizer should be implemented in two or more independent entities simultaneously. In that case an inserted ballot-collection can be detected by comparing the published lists of the different anonymizers.

1.5.3 Submitting a vote

To submit a vote the voter performs the following actions in this order:

1. Have the smartcard generate the VOTER_ID: $MAC_{K_{voter}}(Election\_ID)$.
2. Select the ballot-collection belonging to his VOTER_ID.
3. Choose the desired CANDIDATE_ID.
4. Have the smartcard generate $MAC_{K_{voter}}(Candidate\_ID_{selected})$.
5. Verify $MDC\big(MAC_{K_{voter}}(Candidate\_ID_{selected})\big)$ of the ballot-collection.
6. Submit anonymously the VOTE PAIR: $\big(\text{VOTER\_ID}, MAC_{K_{voter}}(Candidate\_ID_{selected})\big)$

Anyone can calculate $MDC\big(MAC_{K_{voter}}(Candidate\_ID_{selected})\big)$ from this and find out what the value of the vote is. The VOTE PAIR is published by the anonymizer.

1.5.4 Calculating the voting results

After the voting all the received VOTE PAIR's are published. Anyone can calculate the turnout by dividing the number of received votes by the number of published VOTE PAIRs. To calculate the voting results from every published VOTE PAIR the VOTER_ID is looked up in the published table of ballot-collections. The MDC of the second part of the VOTE PAIR is calculated and matched to the chosen candidate in the found ballot-collection. Finally add the votes for each of the candidates together and publish the sum.

1.5.5 Protecting the privacy

The privacy is protected by using an anonymous VOTER_ID which can only be calculated with the help of the smartcard. Unfortunately a submitted vote contains the actual value of the vote in it. Anyone can calculate $MDC(MAC_{K_{voter}}(Candidate\_ID_{selected}))$ from the published vote, thus revealing the vote. This means that if it is possible to link a VOTE PAIR back to the voter, you can reveal which candidate someone voted. To solve this problem we need an anonymous channel with an unknown delay. If the vote is published immediately after reception you can watch who is casting a vote at the moment it is published. This means that there must be a delay between submitting the vote and publishing. A possibility is to queue up the votes at the anonymizer until a certain, large enough number of votes is received before publishing the votes in random

order. To achieve the anonymously the VOTE PAIR may be encrypted with a public-key algorithm using the public key of the anonymizer and submitting the ballot through a number of anonymous gateways. By encrypting the ballot no one but the anonymizer can read the contents of the VOTE PAIR and by using several anonymous gateways the anonymizer is no longer able to determine where the VOTE PAIR was submitted.

1.5.6 Another undesired election property

With a real-time implementation it is possible to calculate temporary election-results on any moment. This offers the ability to verify which candidate receives the most votes at any time. This information may influence the voter. This might be an undesired property, but in some cases it is even wanted. To solve this problem the anonymizer has to publish the VOTE PAIR not real-time but only then when the election is over. This introduces the problem that the voter cannot wait for the publication of his vote to verify that his vote is accepted, which introduces the problem of missing votes. There is no way to be certain that your vote has been counted, unless you can verify it in the public lists. If your vote is not listed you might have not submitted the vote or the anonymizer has silently discarded it to influence the final tally. This could be resolved by introducing several independent anonymizer parties who must agree about the final tally. The vote can be submitted to a certain subset of anonymizers and the anonymizers have to distribute it to all the other anonymizers.

1.6 Threats to the new voting scheme

In this sections we will try to address as much problems as possible in the new system. If possible we will try to address why it is a problem and propose a solution.

1.6.1 Smartcard integrity

The whole system relies on integrity of cryptographic hardware. If the cryptographic keys in the hardware become known the system can be cracked. The most critical functions are implemented at the authority and anonymizer, which is the reason these functions must be implemented in auditable cryptographic hardware. If someone tries to release the keys from the hardware this action is detected and the keys are destroyed. After the voting the hardware can be verified to still work correctly. The major danger lies in the smartcard. Smartcards are subject to attacks since the time they are available. Many amateur hackers have tried to compromise issued smartcards, often with success. This means that any smartcard implementation is suspected to have some weakness by default and a successful attack on the card can't be excluded. If the smartcard is compromised someone may be able to read the keys stored inside and use them to emulate the smartcards functions without the help of the smartcard itself. The key K_{voter} should therefore be unique for each voter. This means that if a single voters smartcard is cracked, only that voter is affected and not the complete system.

1.6.2 Normal DES versus Triple-DES

As described in appendix A the use of the traditional 56-bit DES has became debatable. Because of the increase in computer power any critical implementation that uses DES should use the 112-bit Triple-DES variant.

1.6.3 Time-memory trade-off attack

By publishing the ballot-collections anyone is given access to the MDC-values. A known attack to this publishing is blindly guessing values and calculating the MDC over that value. If the calculated MDC is by chance in the list of published ballot-collections, the attacker can submit the vote connected to that MDC and voter. This means the vote is dependent of which MDC was

found by chance. If we choose the design-parameters large enough we can make the probability of guessing a valid value negligible. This problem is very similar to the problem called "Time-memory trade-off" in the literature [MM82]. Let $r1$ be the number of valid hashes (MDCs) and $r2$ the number of tried guesses. If the hashes are m bit, the number of possible hashes is $R = 2^m$. In this case the probability that no valid hash is found within the $r2$ attempts can be approximated as: $q = (1 - r1/R)^{r2}$. This approximation can only be done when $r2$ is much, much smaller than R and the input values of the hash are statistically independent of each other. If $r1/R \ll 1$ the probability that no value is found can be approximated by: $q \simeq e^{(-r2 \cdot r1/R)}$. For a small probability of finding one of the correct values the following should be achieved: $r1 \cdot r2 \ll R$.

For example if we publish about one million ($r1 = 2^{20}$) hashes of 64 bits each ($R = 2^{64}$), an attacker must try at least $2^{64}/2^{20} = 2^{44}$ possible values to find the input of a listed hash with a reasonable probability. If we can try 200 million hashes per second we will probably find one within a day. The computing power to calculate hashes at such a speed can be achieved at reasonable costs nowadays. We can conclude that the use of a 64-bit hash in large-scale elections is insufficient, instead we should use a larger hash value like 128-bit. The value that the hash is calculated over should be larger than 64-bits as well.

1.6.4 Message tracing

If we need to use a public network to exchange the messages it is possible to determine the sender of a message in many cases. For example if we use the internet to cast our vote a sender address is fixed to that vote. If the sender address can be linked to a person, like in a dialup connection or a PC on someone's desk, it is possible to determine what that person voted. To prevent this public key cryptography is a good solution. The submitted ballot needs to be encrypted with the anonymizers public key, only the anonymizer can decrypt the ballot with his secret key and view what is inside. This function should be implemented with care because the voter is implemented in PC-software which can easily be replaced. Someone could replace the public key of the anonymizer and play a man-in-the-middle attack. Because the authenticity is checked using the MAC calculated in the smartcard only the privacy aspect may pose a problem and only if the anonymizers public key can be replaced. Because the crypto facility is able to perform the RSA algorithm as well, at the anonymizer the decryption should be implemented in that hardware facility which is already needed for the DES decryption. Message tracing is a general problem in public networks.

1.6.5 Message hijacking

If someone is able to reroute the message on its way from the voter to the anonymizer he might be able to discard the message so that the vote will never reach the anonymizer. Altering of the vote is not possible because the MAC's for the other possible votes are never calculated. To prevent the problem of message hijacking, the anonymizer must acknowledge the vote by writing some data (voted message) into the voters smartcard. For a write operation to a field in the card a special key is needed. This key can be generated at the same time as the K_{voter} is generated and should be transported to the card at the registration phase. Multiple anonymizers might be set up to solve this problem. The vote may be submitted at any of those anonymizers and if one fails you can choose another anonymizer and retry the submission until successful.

1.6.6 Compromising the Authority

In the case where the authority is compromised, an attacker could disqualify valid voters from the election by erasing their names from the list of valid voters. Introducing new voters is only possible if those voters have a valid smartcard of which the keys are available to perform the needed operations. The attacker can also set up new elections. This means that the authority can decide who may vote and who may not. The authority should be trusted by all other involved, which is logical because this is the party who initiates the voting. As long as the crypto hardware

is not compromised the authority can't introduce unknown voters without a smartcard. If the cryptographic functions at the authority are correctly implemented with a hardware crypto-facility, compromising the authority doesn't introduce a privacy problem.

1.6.7 Compromising the Anonymizer

If the anonymizer is compromised the attacker is able to introduce new ballots by generating random ballot collections. Because the anonymizer can choose his own input-value to the MDC–function anyone who verifies the final results will think those votes are real. Possible solution is the introduction of several independent anonymizers who verify each other. The introduction of a new ballot-collection will be detected by all other anonymizers. An additional measure is publishing the total number of entitled voters. Anyone can verify if the number of entitled voters equal to the number of ballot-collections. Introducing a new ballot before the election requires the discarding of another from an entitled voter who will complain about his missing ballot-collection. Replacing unused ballots will show up when any individual compares the list of ballot-collections before the elections with the list after the elections have closed.

1.6.8 Compromising the polling booth

The most critical part in this design is the polling booth. Because the polling booth is implemented in (PC)–software an attacker may replace it with a Trojan-horse. A Trojan-horse looks the same as the original software but has a different implementation which may be malicious. An attacker can easily mislead the voter by indicating that it votes for candidate A, but in the background have the smartcard calculate the MAC for candidate B and submitting that. Another problem is that if the smartcard is inserted in the smartcard reader any data that is on the card can be read. On almost any card personal information like name, student-number or account-number is publically available. Although that personal information is not needed to complete the voting any malicious program can read that information and use it to link the person to the selected vote. Solving this problem is very hard and during implementation special care needs to taken in regard to this subject. A possible solution is that every political party releases its own voting-software and you can choose the software from someone you trust, you own party for example. No one will ever release software that fakes the user in such a way that votes for his own party gets lost.

1.7 Evaluation of the requirements

Before describing the design we listed some requirements that apply to votings. We will now verify if the proposed scheme satisfies those requirements by describing how each requirement is satistfied:

Only authorized voters can vote: This is true because ballot-collections are only available for entitled voters.

No one can vote more than once: This is true because the authority generates only one ballot-collection for each voter.

No one can determine for whom anyone has voted: This is true because the votes are published with an anonymous VOTER_ID. The VOTER_ID can't be linked to a person, thus the vote can't be linked to a person.

No one can duplicate anyone else's vote: This is true. Duplicating the MAC from a known VOTE PAIR is not useful because the MAC for a specific vote is different for each voter.

No one can change anyone else's vote without being discovered: This is true because no one else but the voter can calculate a new MAC for a different candidate. Changing the vote would require calculating a new MAC.

Every voter can make sure that his vote has been taken into account: This is true because all the ballots are published the results can be recalculated by the voter. The voters can verify if their votes are counted correctly by viewing the vote stated with their own VOTER_ID.

Everyone knows who voted and who did not: This requirement is optional and partly satisfied. Anyone can calculate the turnout but not who voted and didn't vote.

The additional non-coercion requirement can be satisfied only partly. In a normal implementation the smartcard can calculate the VOTER_ID as many times as you like, giving the possibility of proving a vote by showing someone your VOTER_ID. By implementing the function like an electronic purse, in which money can be spent only once, this problem can be solved. The calculation of VOTER_ID is possible only once or twice. This does not solve the non-coercion requirement completely, because the voter knows information that is only known to the voter and the anonymizer before it is published by the anonymizer. A voter can prove his vote by showing that information before it is published. If the given information is published by the anonymizer the vote is proved because only the voter himself has knowledge of that information.

1.8 Diagram of the new voting protocol

A diagram of the new voting-protocol is given in figure 1.3.

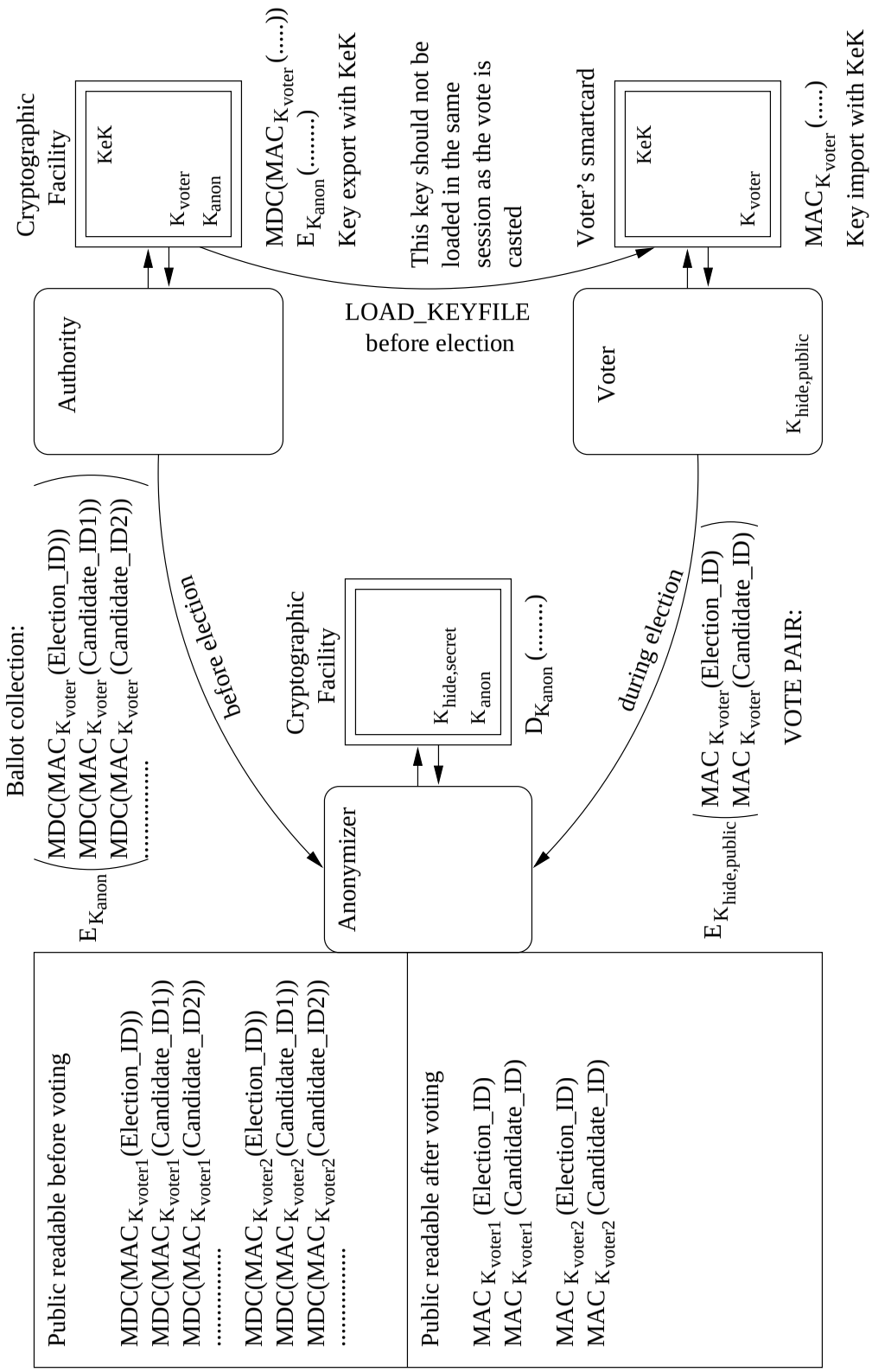

Figure 1.3: Diagram of the new secure electronic voting protocol

Chapter 2

Conclusions

In this paper a theoretical design is given for a voting protocol that uses current technology smartcards and periphery. Everything described could be implemented in a secure way with today's commercially available products. The main advantage of this new design is that contrary to other secure voting schemes for critical operations, symmetric cryptography like DES can be used. Because the cryptographic operations in this system do not rely on certain properties of the DES algorithm not available in any other algorithm, asymmetric protocols like RSA can be used as well. This makes the designed protocol more flexible than other proposed protocols.

Another outcome is that a new election requirement is defined: The voters may not be able to view the election results before the elections have closed. This is needed to prevent influencing the voters who didn't vote yet.

2.1 Recommendations

The design of this protocol can be implemented by a successor graduate student and one or two trainees. Getting familiar with the cryptographic hardware will probably require several months and implementation of the required cryptographic functions will be a full internship job. In a prototype implementation the application on the side of the voter does not need to be perfect but a commercial version will require a lot of attention. To make the voting accessible to all voters the software needs to be extremely user-friendly and a lot of effort should be put in ergonomics, usability and trust of the system.

Glossary

CHV: Card Holder Verification. Also known as PIN or numberlock. A code that must be supplied to the card to show that you are the owner of the card. In most cases this is a 4 digit number.

Control vectors: A method invented by IBM to limit the functionality of hardware cryptographic solutions to only the most nescessary functions. A certain key can be given the property to perform only certain operations, like encryption only or MAC verification only. If used in a safe way this gives asymetric properties to a symetric algorithm like DES. The IBM smartcards use a limited set of control-vectors to prevent certain attacks.

DES: Data Encryption Standard. A symmetric cryptographic algorithm dealing with 64–bit blocks of data and a 56–bit key. Triple DES uses two 56–bit keys making the algorithm theoretically unbreakable. See appendix A for a description of DES.

ISCIT: IBM Studenten Chipkaart Innovatie Team, or IBM Student Chipcard Innovation Team. A team of students graduating or doing their internship on new smartcard technologies within IBM Netherlands N.V.

MAC: Message Authentication Code. A derivate of DES implementing a one-way hash function. In general the MAC is used to create a signature over a datafield to protect both integitry as well as authenticity. See appendix A.3 for a description of MAC.

MDC: Modification Detection Code. A one-way function developed by Carl Meyer and Michael Schilling used in the IBM TSS cryptosystem.

non-coercion: The requirement that a voter can not prove his vote. This is important in selling and buying of votes.

SCK: Studenten Chipkaart. The chipcard developed at ISCIT distributed by some Dutch educational institutes.

Appendix A

Data Encryption Standard

The symetric encryption algorithm DES (Data Encryption Standard) was developed in the 70's as proposal to the American government departing from IBM's Lucifer cryptoalgorithm. On May 15, 1973 [MM82], the National Bureau of Standards (NBS) published a notice in which it asked for proposals for cryptographic algorithms. According to the NBS, the algorithms should live up to the next points:

1. They must be completely specified and unambiguous.
2. They must provide a known level of protection, normally expressed in length of time or number of operations required to cover the key in terms of the perceived threat.
3. They must have methods of protection based only on the secrecy of the keys.
4. They must not discriminate against any user or supplier.

According to the NBS, only one entrance submitted (by IBM) was found acceptable. This algorithm later became known as the "Data Encryption Standard" (DES). DES is *the* standard on Secret-Key algorithms.

DES encrypts data in 64-bit blocks (using the block ciphering method). Both the input block and the output block are 64-bit. The length of the key is 56 bit. This key is actually 64 bits long, but the last 8 bits are used for parity. The steps DES performs, after the initialisation (the initial permutation), at each block-encipher round (DES has 16 rounds) are the following ([MM82]):

1. The input block is split into two parts; a left half and a right half.
2. The right half (step 1) is then operated using a cipher-function.
3. This output (step 2) is combined (via an XOR) with the left half.

After 16 rounds, the right and left halves are joined and a final permutation (which is the inverse of the initial permutation) completes the algorithm.

A.1 Security of DES

Since the publication of DES many efforts have attempted to break the algorithm. Many believed there should be a backdoor for the government to bypass the algorithm. Until today, more than 20 years after publication, not a single backdoor has been found. Recently methods using differential cryptanalsis have reduced the effort to find a DES-key, given you can perform well chosen plaintexts and have them encrypted. Most likely to exploit is the brute-force attack on a pliantext-ciphertext pair, because DES can be implemented in a very efficient way. Attacking DES with a brute-force attack is nothing else than trying all possible keys on a given plain- or ciphertext and check if

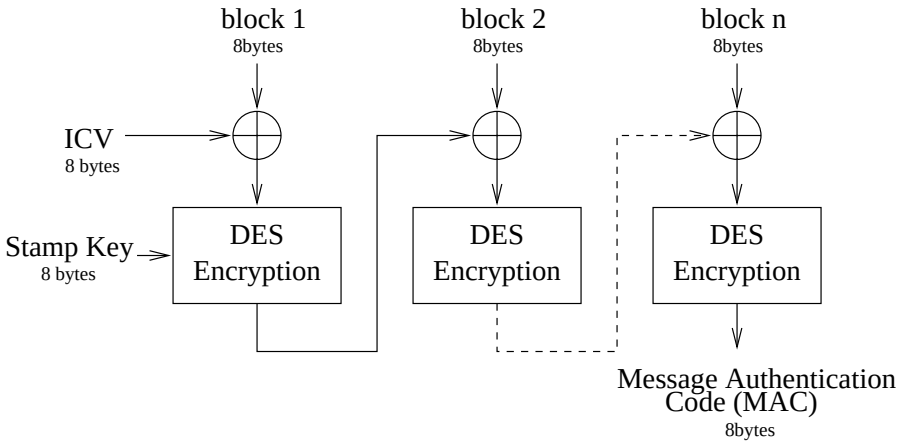

Figure A.1: The X9.9 Message Authentication Code (MAC)

the output is the one you were searching for. The July 1998 RSA labs DES-challenge, a contest cracking DES is the shortest possible time, was finished in 2.3 days by a projectgroup that built a hardware DES-checker. Total expense of the project was under $ 250,000. Their machine found the key at a quarter of the keyspace, which means it could check all possible 56-bit keys in 9 days. This means that any cryptosystem that makes use of DES and reveals plaintext-ciphertext pairs can be cracked within a short time with a resonable amount of money. This is the reason that heavily secured processes can't use 56-bit DES for its protection. All IBM-systems that use symetric cryptography use triple-des by default since 1978.

A.2 Triple DES

Triple DES is an expansion of the existing 56-bit DES, and uses 2 56-bits keys making the total keyspace 112 bits. The triple in Triple-DES states that it uses three standard DES-operations: one encryption, one decryption with another key and again an encryption with the first key. Note that if both keys are equal a normal DES-operation appears. The first encryption and the decryption cancel out eachother. It is believed to be computationally infeasable to brute-force attack Triple-DES. Most financial transactions and encryption of PIN's are done using Triple-DES.

A.3 DES Message Authentication Code (MAC)

The MAC uses DES in Cipher Block Chaining mode. Cipher Block Chaining mode is a mode of DES where the data that must be encrypted is chopped in 8 bytes blocks and the result of a DES-encryption is part of the input of the next step. A schematic overview of a MAC calculation is shown in Figure A.1. The value at the end of the chain is called the MAC. Because all datablocks used in DES are 8 bytes the MAC is 8 bytes as well. The MAC depends on both the data the MAC was calculated over as well as the stampkey. A MAC can be used to secure the transportation of a message, because if the message is changed on its way the MAC no longer matches the message. Because the the stampkey is used in the MAC, it can also be used to check authenticity. Only when you know the stampkey you can calculate the correct MAC. Since DES is a symetric algorithm you need to have the stampkey to verify the MAC and thus you can't prove which of the parties that know the key actually signed the message.

In fact you could have signed it yourself. The use of reliable cryptographic hardware could solve that problem. This type of MAC is published in the Banking Standard X9.9. A triple DES-variant is published as X9.19, in which only the last DES encryption is changed into a Triple DES

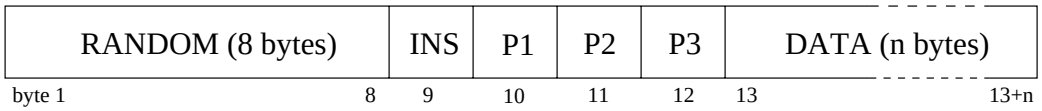

Figure A.2: Data in the MAC caclulation of the IBM MFC-3.51 smartcard

encryption. This way the property of Triple-DES appears where you choose the 2 keys equally it reduces to a single DES-MAC.

The MAC calculated in the IBM MultiFunctionCard uses zero as input vector (i.e. all bits 0) and as data a composition of random, command (INS, P1, P2 and P3) and returned data as shown in figure A.3. The *INS* byte always has value `B4` (hex), *P1* and *P2* represent the offset in the file and *P3* is the number of bytes to be read.

A.4 Control Vectors (CV)

Control Vectors is a system invented by IBM and implemented in all cryptographic hardware devices [IBM98] of that company. Control Vectors can limit the allowed operation performed with a key. By using control vectors the symetric cryptography is given asymetric properties, given that the operations are performed within the cryptofacility. For example some key can be given the property to only allow encryption with that key, if in another similar system the same key is available with the control vector set to allow decryption only a separation of functions is possible. Many designs use the property of function separation to implement a safe protocol in which one party can only perform the opposite action of the other.

The control-vector is a key-like value (the same length as the master key) describing which functions the hardware module may perform in combination with some key. Before the key is used it is XOR–red with the controlvector and then decrypted with the Key Encrypting Key (KEK) resulting in the desired working key. Before performing the requested operation the hardwaremodule verifies if that specific operation is allowed according to the used controlvector. If we would try to fool the hardwaremodule by offering another controlvector, which allows operations we need to crack the system, the calculation of the working-key fails because the input is dependent on both the control-vector as well as the encrypted key as shown in figure A.3. Some oparations that can be controlled with contol-vectors are:

- CIPHER: This key can be used for encryption
- DECIPHER: This key can be used for decryption
- MAC: This key can be used to generate a MAC
- MACVER: This key can be used to verify a MAC

Much more operations can be defined. For a full explanation of control vectors see [IBM98]

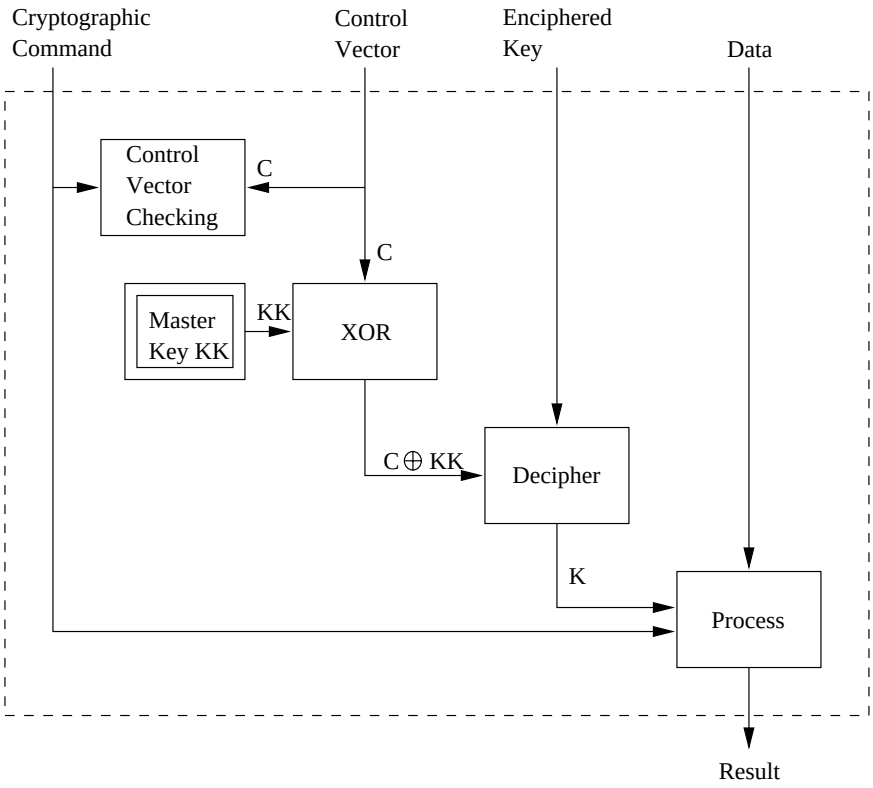

Figure A.3: Operation with a derived key in IBM cryptographic hardware

Appendix B

Smartcards and authentication

The smartcard was invented in 1974 by Roland Moreno, he invented a ring with electronics that could be used as the first known electronic purse. You can transfer money to your ring and pay with it at a special device at the grocery. In the early eighties the smartcard became more widely spread. The French postal service introduced a memory card to pay at public phones, shortly after that the more advanced microprocessor card was introduced. What makes this processor card special is that current cards contain a processor with about the computing power of the first personal computers. It might not be a surprise that this allowed great new applications. The best known task of the microprocessor is to perform cryptographic computations. This can be used for creating secure applications like banking and remote authentication.

If a smartcard or chipcard is mentioned in this document, the microprocessorcard is meant. The terms chipcard and smartcard are used interchangeable.

B.1 Dutch Students Chipcard

The card I worked with is the Dutch Students Chipcard (In Dutch: Studenten Chipkaart, abbreviated SCK). This card is issued by the foundation SCK and supplied to 150,000 students in 1998. Issuing the card to this critical public of students was done to detect problems in large scale chipcard projects. As a bonus some students have fun with searching the card for weaknesses and at this pilot stage it is possible to make adaption to the card design before a issuing a huge roll-out.

The card used in the Studentchipcard project is an IBM MultiFunctionCard version 3.51. This card employs the symmetric cryptographic DES-algorithm. The new MFC 4.0 card can also perform the asymmetric RSA algorithm, but this card is not available in large quantities yet, so we will try to use the characteristics of the symmetric MFC 3.51 card as much as possible.

B.2 Authentication with the MFC

Authentication is the process that determines if a message is really sent by the person who says he is. It also detects altering of the message or the authentication because they need to match.

The MFC card has three standard methods for authentication:

- Encryption (see figure B.1). In this method the card performs an encryption of a given value M with a key K available on the card. The results, $E_K(M)$ are returned to the requester. By decrypting the returned value with the same key K the given value M should appear. In that case you are sure about the possession of key K without exchanging that key. This method authenticates the card to the outside world.

- Protected (PRO) (see figure B.2). Some data on the card is read and a MAC using a key K is added to provide authenticity. The requesting party generates a random value an sends

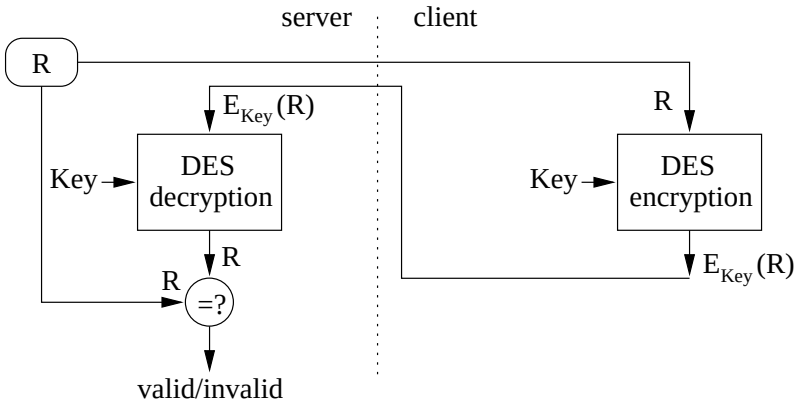

Figure B.1: Authentication using encryption

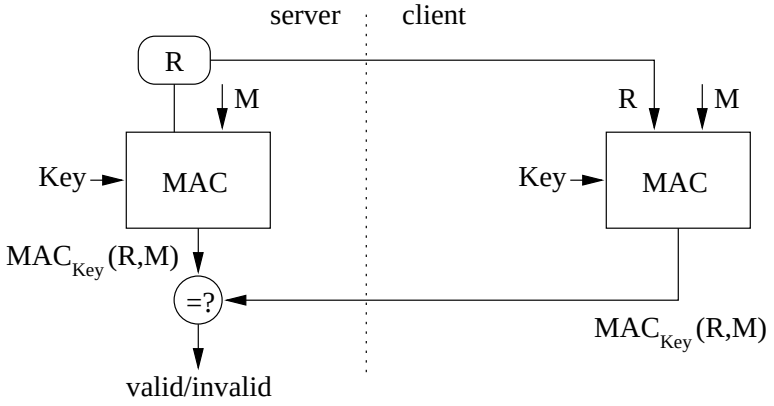

Figure B.2: Authentication using a MAC

it to the card. This value is used to make the MAC-value dynamic. This method also authenticates the card to the outside world.

- Authenticated (AUT): This method is the opposite of PRO. Now, the card generates a random value and the command to the card must be accompanied by a MAC of that random value and the command. This method authenticates the outside world to the card.

We should note that the use of the encryption authentication method is a bad thing in general, because this releases plaintext-ciphertext pairs. This means that an attacker can collect the plaintext and the according ciphertext. Because it is known that the ciphertext is only a DES-encryption of the plaintext a dedicated hardware cracker can be used to brute-force try all the keys and find the used key. This authentication is cracked when the key is found. Because an attacker with possession of the card can send carefully chosen plaintext and gain the according ciphertext some more efficient attacks are possible. So in practice only AUT and PRO can be used safely. The MAC-calculation is slightly more complicated and additional data is used. No standard hardware is available to perform an efficient brute-force attack. The major disadvantage of the implemented authentication function in standard ETSI TE9 and many other chipcard standards is that the message M is transferred between both systems in the clear. The birthday attack [MM82] applies in this case. Proper authentication protocols for DES that do not suffer from these weak properties have been designed [MM82] (see the "session protection protocol") but the designers of the popular authentication functions in the chipcard world apparently were not familiar with that.

Bibliography

[CFSY96] Ronald Cramer, Matthew Franklin, Berry Schoenmakers, and Moti Yung. Multi-authority secret-ballot elections with linear work. In Ueli Maurer, editor, *Proceedings of EUROCRYPT'96*, volume 1070 of *Lecture Notes in Computer Science*, pages 72–83. Springer-Verlag, 1996.

[Cha83] D. Chaum. Blind signatures for untraceable payments. *Advances in Cryptology Crypto '82*, pages 199–203, 1983.

[Cra96] Lorrie Faith Cranor. Design and Implementation of a Practical Security-Conscious Electronic Polling System. Technical Report WUCS-96-02, Washington University, Department of Computer Science, January 1996. Available for download at http://www.ccrc.wustl.edu/~lorracks/sensus/.

[FOO92] A. Fujioka, T. Okamoto, and K. Ohta. A practical secret voting scheme for large scale elections. In J. Seberry and Y. Zheng, editors, *Advances in Cryptology - AUSCRYPT '92*, volume 718 of *Lecture Notes in Computer Science*, pages 405–419. Springer-Verlag, 1992.

[IBM96] IBM. *IBM Multi Function Card – Programmer's Reference for Version 3.51 including extensions EE20.* IBM Smart Consumer Services, November 1996. IBM CONFIDENTIAL.

[IBM98] IBM 4758 CCA Basic Services Reference and Guide. http://www.ibm.com/security/cryptocards/, 1998.

[MM82] Carl H. Meyer and Stephan M. Matyas. *Cryptography – A new Dimension in Computer Data Security.* John Wiley and Sons Inc., 1982.

[Rob98] H.W.K. Robers. HTTP Authentication using smartcards. Technical report, IBM Student Chipcard Innovation Team, June 1998.

[RRB97] R.Cramer, R.Gennaro, and B.Schoenmakers. A secure and optimally efficient multi-authority election scheme. In *Proceedings of EUROCRYPT'97*, volume 1233 of *Lecture Notes in Computer Science*, pages 103–118. Springer-Verlag, 1997.

[Sch96] B. Schneier. *Applied Cryptography.* John Wiley and Sons Inc., 1996.

[vdL97] T. van der Laan. Architectuur voor project: Studentenchipkaart 1997. DUTCH, Internal pre–release, march 1997.