Anonymous Code Lists For Secure Electronic Voting Over Insecure Mobile Channels

Nico Voutsis
Hewlett-Packard
In der Luberzen 29
CH 8902
Urdorf, Switzerland
Email: nico.voutsis@hp.com
http://www.hp.com/

Frank Zimmermann
Hewlett-Packard
In der Luberzen 29,
CH 8902
Urdorf, Switzerland
frank.zimmermann@hp.com
http://www.hp.com/

Abstract: *A protocol is proposed, which allows electronic voting over channels, which typically are regarded as insecure like short message service (SMS) and cellular phones. Unlike personalized lists of codes, which are typically proposed to secure the vote and to uniquely identify the voter, we propose anonymous code lists, which offer more flexibility and security, in particular with respect to privacy and anonymity of the voter.*

Keywords: e-voting, mobile voting, security protocol, m-government.

1. Introduction

Back in January 2003, the state of Geneva, Switzerland, performed the first official e-voting over the internet. In the period before and during the ballot period and also in the subsequent ballots, the solution went through a thorough security analysis and extensive testing (Geneva, 2005). Due to the intense spread of cellular phones and other mobile appliances for data communication, the additional requirement arose, to also enable voting over these channels, which typically are regarded as insecure. This requirement motivated us to develop a communication process and algorithm for "secure electronic voting" over "insecure mobile channels" based on "anonymous code lists".

What is e-voting? The so-called e-voting denotes any mechanism to cast a vote or to participate in elections, by which the relevant data are transmitted over a network (internet or mobile network). In most cases, e-voting represents an additional voting channel, which is not mandatory to use. It underlies the same requirements with respect to security as the conventional voting and election procedures at the poll site or for postal voting (see (Warynski, 2003), (Zimmermann, 2003) and references therein). An extended version of this paper will be published soon in a book. There, we will also describe in more detail the most critical security requirements and how to address them. In this shorter paper, we will focus on the key elements of the proposed protocol.

For any (technical) solution, the concurrent requirements of authorization and anonymity are the most challenging: the permissions of the voter must be controlled, it must be assured that the voter just votes once, and it is possible to prove that a given voter has voted, but it must not be possible to associate the content of any vote with a voter. This implies a total separation of the registry of voters from the electronic ballot with the votes and the tallying of the votes. A chronological separation into four distinguished voting processes is necessary in order to achieve this and other security requirements (figure 1):

a. Initialization phase: generation of voter identification codes and code tables, printing of the anonymous code lists with the ballot papers and the voter cards, initialization of the voter registry and the electronic ballot,

b. Voting phase: voting or election on the different channels (electronic, mobile, postal, or at the polling station),
c. Tallying phase: tallying of the votes that were cast through the different channels and consolidation of the results, and
d. Cleanup phase: cleanup of the systems, voter and vote registries.

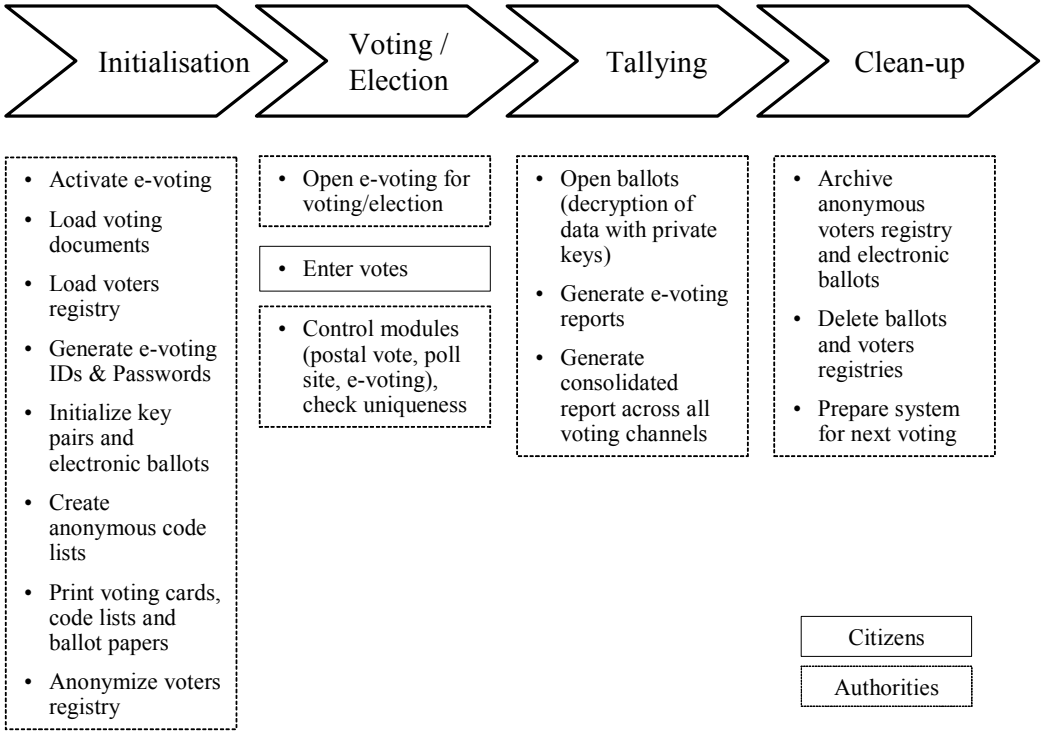

Figure 1: Strict separation of 4 distinguished voting process phases and corresponding sub-processes.

2. Anonymous code lists in the context of e-voting

In the following we focus on the description and discussion of the algorithm for the generation and usage of the anonymous code lists. Because of the importance of the separation of the different voting phases, this section is structured along the above mentioned phases for electronic voting.

2.1. Initialization

During the initialization phase, the anonymous code lists and the voter's credentials for the participation in the ballot are generated and distributed. The voter cards are the identification instrument of the voter. Usually, they provide the voter's personal credentials for all the different voting channels: poll site, postal and electronic. For e-voting, for instance, an identification code together with an additional password could be used, which can only be used once and is hidden behind a "scratch field" like it is used in lottery scratch tickets (figure 2). Other implementations are printing unique barcodes on the voter cards, such that it can easily be checked against an online registry at poll site or in the postal voting, if voters already voted via another channel. A sophisticated solution could use electronic identity cards with digital certificates.

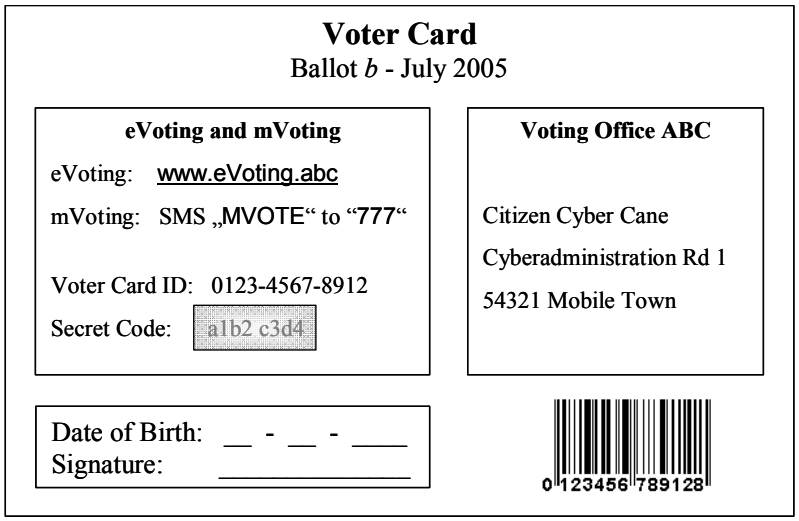

Figure 2: Typical voter card.

Due to the anonymity requirement the ballot papers and code lists, which are described below do not hold any personal information or any association to a voter. Of course, the voter should only receive a ballot paper with a poll, that she or he is authorized to participate in. Figure 3 provides an overview of the algorithmic description of the protocol during the initialization phase, augmented by some examples. The details of the various phases will be further outlined in the following chapters.

Ballot paper with poll questions and possible answers	**Anonymization** by randomized variation of possible answers with obscurity factor L	**Encryption** with >2 independent public keys to avoid reconstruction of answers during voting phase	**Code generation & assignment** with unique, but easier to use codes stored in code table	**Printing of Ballot paper with code lists** and deletion of any links between codes and answers
Poll questions $k\ (k=1,...,K)$: $q_1..q_K$ Choices of answers to q_k: $a_k^j \in A_k, j=1,...,\lvert A_k\rvert$ Set of all choices of answers: $H=\bigcup_k A_k=\{h_i\}, i=1,...,\lvert H\rvert$	Obscurity factor L: $\forall i=1,...,\lvert H\rvert, l=1,...,L$ $h_i \mapsto \{g_i^l\}=\{(i,r_i^l)\}$, r_i^l = random up to γ bits	Encryption f $\forall i=1,...,\lvert H\rvert,\ l=1,...,L:$ $g_i^l \mapsto e_i^l = f(g_i^l)$ where: $e=f(g)=f_2\circ f_1(g)$ $=\left(g^{\varepsilon} \bmod(n_1)\right)^{\varepsilon} \bmod(n_2)$ with RSA public keys $f_1:(n_1,\varepsilon), f_2:(n_2,\varepsilon)$ Default exponent: $\varepsilon=65537$	Code table: $\forall i=1,...,\lvert H\rvert,\ l=1,...,L:$ $T=\{(e_i^l \leftrightarrow c_i^l)\}$ with: $c_i^l=(s_i^l,b_i^l)$, s_i^l = random string b_i^l = control character	Code list $(m=1,...,M)$: Poll questions $k\ (k=1,...,K)$: $q_1..q_K$ Choices of answers: $a_k^j \in A_k, j=1,...,\lvert A_k\rvert$ $q_k: a_k^j \mapsto (a_k^j, c_i^l)$ $l \in \{1,...,L\}$ random for each m, and i defined by $i: h_i = a_k^j$.
Example: **Ballot paper** Ballot b – July 2005 Voting District XYZ Poll question 1: q_1 = *"Joint the EU ?"* Choices of answers: O a_1^1 = *"yes"* O a_1^2 = *"no"* O a_1^3 = *"blank"*	Example: $h_2 = "no" \Rightarrow \quad i=2$ $h_2 \mapsto \{g_2^1...g_2^L\}$ $g_2^1 = (\underset{0}{\lfloor} A97...1F0 \underset{56}{\rfloor} 0002 \underset{72}{\lfloor} 2C3...854 \underset{\gamma}{\rfloor})$ random i random $g_2^L = (\underset{0}{\lfloor} 5E2...D03 \underset{56}{\rfloor} 0002 \underset{72}{\lfloor} B86...491 \underset{\gamma}{\rfloor})$ in hexagesimal code, e.g. with $\gamma=512$. $\{g_2^1...g_2^L\} \mapsto \{e_2^1...e_2^L\}$ $e_2^1 = (\underset{0}{\lfloor} 93B...6A2 \underset{\delta}{\rfloor})$ $\vdots \quad \cdots \quad \vdots$ $e_2^L = (\underset{0}{\lfloor} 0E2...491 \underset{\delta}{\rfloor})$ Default: $\delta=1024$.		Example: $(c_2^l \leftrightarrow e_2^l)$ with $e_2^l = (\underset{0}{\lfloor} 93B...6A2 \underset{\delta}{\rfloor})$ c_2^l = "2zw75n".	Example: Print & delete! **Ballot paper & code list** Ballot b – July 2005 Voting District XYZ Poll question 1: q_1 = *"Join the EU ?"* Choices of answers: O a_1^1 = *"yes"* n83k2q O a_1^2 = *"no"* 2zw75n O a_1^3 = *"blank"* s953mb

Figure 3: Overview: initialization phase.

The code lists can be directly printed on the ballot papers, but could also be printed and distributed independently (figure 4 and figure 5). Exactly one ballot paper is distributed for each voter. The ballot papers, code lists and the voter cards are printed independently. With anonymous code lists – by default – there is not any association between the code list and the voter, which could be used to identify a voters vote. The process of assembling and distributing the voting documents, (voter card, code list, and ballot paper) becomes relatively simple, because only the voter card contains personal information and ballot papers and code lists can be mixed randomly. For even better randomness the generated code lists are shuffled randomly before they are printed and distributed together with the voter cards and the ballot papers. The following paragraphs describe the creation of the code lists in more detail.

2.1.1. Ballot Paper

Figure 4 shows a typical example of a ballot paper (still without any codes). For each poll question q_k there is a set A_k of choices of answers a_k^j, with j numbering the answers.

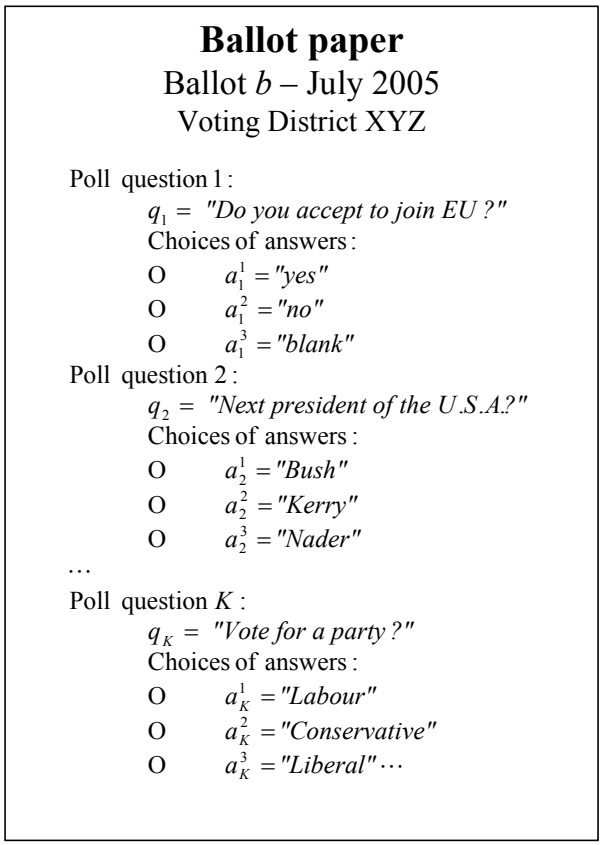

Figure 4: Ballot paper without codes.

The set H of all possible choices of answers in a poll is defined by the sum of all answers to each poll questions. Be $|H|$ the number of all possible choices of answers H in a ballot b, then in our example, we get:

Ballot paper for ballot *b*

Poll questions k ($k = 1,...,K$): $q_1..q_K$, e.g.

$q_1 =$ "Do you accept the proposal to join EU ?"

$q_2 =$ "Next president ?"

Choices of answers to q_k: $a_k^j \in A_k$, $j = 1,...,|A_k|$, e.g.

$A_1 = \{"yes","no","blank"\}$

$A_2 = \{"Bush","Kerry","Nader",...\}$

Set of all choices of Ballot: $h_i \in H,\ i = 1,...,|H|$

$$H = \bigcup_k A_k = \{"yes","no","blank","Bush","Kerry","Nader",...\} = \{h_i\}$$

If there are several questions q_x that can be answered with *"yes"* and *"no"*, we recommend to include each occurrence of *"yes"* and *"no"* separately in the set of choices. Having this 1:1-relationship between a question and its set of choices, each choice (i.e. vote) can be directly associated to the according question during the tallying process. The introduction of H, h_i and i simplifies the arithmetic and underlying processes.

2.1.2. Anonymization and encryption

The possible vote answers h_i (e.g. *"yes"*, *"no"*, or *"blank"*) are then represented by a bit-pattern. The simplest way to do so is to index the set of choices and to take the binary or hexadecimal representation of the index. This representation would then look as follows:

Examples			
Choice of answers	index	binary	hexadecimal
"yes"	*1*	*0000 0001*	*01*
"no"	*2*	*0000 0010*	*02*
"blank"	*3*	*0000 0011*	*03*
...	...	...	
"uvw"	*11*	*0000 1011*	*0B*
…	…	…	
"xyz"	*109*	*0110 1101*	*6D*

Of course, more complex bit representations are applicable. The number of bits can either be fix for all ballots (e.g. *16* bits allow for $2^{16} - 1 = 65535$ possible choices) or vary for each ballot depending on the size of the actual set of choices. We call these the binary and hexadecimal representation of the vote also bit representations of the vote.

The bit representations of possible vote answers h_i are then anonymized by adding random bits to the bit representation of the vote and the result is encrypt with the public keys of authorities (for cryptographic algorithms like RSA public key encryption see (Schneier; 1996)). Of course, the authorities have to possess the corresponding private keys as well in a highly protected manner (e.g. on smart card or other tokens). We suggest that at least two independent authorities provide their public encryption keys for the encryption. This ensures that no single authority is capable of decrypting the cast votes that are stored in the electronic ballot box. Note: there also exist more sophisticated algorithms (e.g. *homomorphic* encryption), which involve more than two authorities and corresponding public keys, which only require the presence of a minimal set of authorities being present with their corresponding private keys for the tallying (Hirt, 2001).

Repeat this step L times for each of the $|H|$ vote representations. The obscurity factor L is a large number, ideally (but not necessarily) as large as the number of eligible voters M (the size of L is discussed below).

Randomized encryption of vote choices

Obscurity factor L :

$$\forall i = 1,...,|H|,\ l = 1,...,L : h_i \mapsto \{g_i^l\} = \{(i, r_i^l)\},$$

where r_i^l = random bits up to γ bits.

Encryption f with two or more independent public keys

$$\forall i = 1,...,|H|,\ l = 1,...,L : g_i^l \mapsto e_i^l = f(g_i^l)$$

Using two RSA public keys

$$e = f(g) = f_2 \circ f_1(g) = \left(g^{\varepsilon} \bmod(n_1)\right)^{\varepsilon} \bmod(n_2)$$

with public keys $f_1 : (n_1, \varepsilon), f_2 : (n_2, \varepsilon)$

and default exponent : $\varepsilon = 65537$

There are several ways to group the random bits around the vote representation bits, e.g., random bits up to bit 56, 16 bits for the vote representation, and the rest of the bits are random again. This process turns one simple answer *"yes"* or *"no", "Kerry"* or *"Bush"* into multiple random bit representations that represent the same information *"yes"*, *"no"* etc.

Example :

$h_2 = "no" \Rightarrow \quad i = 2$

Randomize :

$$h_2 \mapsto \{g_2^1 ... g_2^L\}$$

$$g_2^1 = (\underset{0}{|}\ A97...1F0\ \underset{56}{|}\ 0002\ \underset{72}{|}\ 2C3...854\ \underset{\gamma}{|})$$

random i random

$$g_2^L = (\underset{0}{|}\ 5E2...D03\ \underset{56}{|}\ 0002\ \underset{72}{|}\ B86...491\ \underset{\gamma}{|})$$

in hexagesimal code, e.g. with $\gamma = 512$ or 1024.

Encrypt :

$$\{g_2^1 ... g_2^L\} \mapsto \{e_2^1 ... e_2^L\}$$

$$e_2^1 = (\underset{0}{|}\ 93B...6A2\ \underset{\delta}{|})$$

$$\vdots \quad \cdots \quad \vdots$$

$$e_2^L = (\underset{0}{|}\ 0E2...491\ \underset{\delta}{|})$$

with default : $\delta = 1024$.

2.1.3. Code generation and assignment

Each encrypted randomized vote answer is mapped to a unique random string of characters and Hamming codes (control character(s)) for error detection are added, accordingly. The resulting unique 1:1 mapping is stored into a code table T for later usage in the tallying process.

Code generation & assignment

Code table generation & code assignment

$$\forall i = 1,...,|H|,\ l = 1,...,L : e_i^l \mapsto c_i^l = (s_i^l, b_i^l),$$

where s_i^l = unique random string of characters

e.g.. 5 or 7 characters in $P = \{2,...,9,a,...,z\}, p = |P|,$

and an extra b_i^l = control character

Store all $(e,c) \in T$ in table $T = \{(e_i^l \leftrightarrow c_i^l) : \forall\ i = 1,...,|H|,\ l = 1,...,L\}$.

Example :

$$(c_2^l \leftrightarrow e_2^l) \quad \text{with} \quad e_2^l = (\overset{|}{\underset{0}{|}} 93B...6A2 \overset{|}{\underset{\delta}{|}}) \text{ and } c_2^l = \text{"2zw75n"}.$$

Each of the M eligible voters has to get a different code list, in order to re-assure that every voter has one and only one vote. Therefore, ideally, the obscurity factor L is selected such that $L=M$. On the other hand, codes with more than 5 or 7 characters might not be user-friendly and error-prone. This will limit the total number of possible unique codes to $L|H|<p^5$ *or* p^7, i.e. the number of variations of 5 or 7 out of p different characters with repetition. If codes with the $p=32$ distinguishable characters are used in an election with maximal $|H| = 128$ different poll choices, then more than 250'000 eligible voters could receive code lists with unique codes, which are 6 characters long (incl. control character).

Examples				
# of answers $\|H\|$	# characters in code string p	code length	maximal L	Typical case
128	*32*	*5+1*	*266'144*	State of Geneva
128	*32*	*7+1*	*268'432'456*	US President
1024	*32*	*7+1*	*33'554'432*	Parliament Election

In order to limit the code length to user-friendly 5+1 characters, one might be forced to accept an obscurity factor L smaller than M, the number of eligible voters. Thus, one and the same code could occur on different code lists and therefore could be used by different voters. In that case, the unique code list number $m=1,...,M$ must be recorded and also validated in the voting phase in junction with the codes, in order to distinguish false double votes from the eligible usage of the same code from different code lists. A user-friendly choice will be a code length of 5+1 or 7+1 characters (note that the code list number also easily gets 9 digits long).

A remark on random number generation: it is very important to select pseudo random number generators with a reasonably large period and which pass all known tests for randomness. A random number generator of best mathematically proven quality is the one proposed by Lüscher (Lüscher, 1993), which is based on the *subtract-with-borrow* generator (Marsaglia & Zaman, 1991) and has a period of approximately 10^{171}. If only a relative small - but unique - sequence of random bits is required, then taking the leading bits of a random sequence with a smaller period will be sufficient. For instance, a *linear congruential* generator can be taken: $x_{n+1}=261547876541325\ x_n \bmod 2^{48}$ (with a period of $2^{46} \approx 7\ 10^{13}$, see: (Masuda & Zimmermann, 1996) and references therein). Since pseudo random number generators are periodic and generated according to a specific rule, the random seeds used should never be made public – it should also not be hard coded in the voting software, which may have to be published according to the voting regulations.

2.1.4. Generation and printing of ballot papers with code list

The M anonymized code lists (for M eligible voters) are generated by randomly assigning appropriate codes from the above generated code table to corresponding vote answers (figure 5):

Shuffle codes to code lists / ballot papers

Code list m $(m = 1,...,M)$ for ballot b:

Poll questions: $q_1..q_K$ $(k = 1,...,K)$ and corresponding answers $a_k^j \in A_k$ $(j = 1,...,|A_k|)$:

$q_k : a_k^j \mapsto (a_k^j, c_i^l)$ where $l \in \{1,...,L\}$ selected randomly, i defined by $i : h_i = a_k^j$;

$q_{k_1} \neq q_{k_2} \Rightarrow c_{i_1}^{l_1} \neq c_{i_2}^{l_2} \forall l_1, i_1, l_2, i_2$

Example:

$q_1 =$ "Do you accept the proposal to join EU ?"

$\{(yes, \texttt{"n83k2q"}), (no, \texttt{"2zw75n"}), (blank, \texttt{"s953mb"})\}$

The code list (i.e. ballot papers with codes) are printed as generated, and any information is deleted, which could relate vote answers (*"yes"*, *"no"*, etc.) with codes from the code table. Once this information is deleted from the systems, the codes can only be deciphered in the tallying phase by means of the corresponding private keys of the authorities mentioned above. Thus the codes in the code lists can be used in the voting phase for secure voting over insecure channels, without any link of the codes to the respective answers except the print-out on the ballot paper or code lists.

Ballot paper

Ballot *b* – July 2005

Voting District XYZ

Poll question 1:

$q_1 =$ *"Do you accept to join EU ?"*

Choices of answers:

O $a_1^1 =$ *"yes"* `n83k2q`

O $a_1^2 =$ *"no"* `2zw75n`

O $a_1^3 =$ *"blank"* `s953mb`

Poll question 2:

$q_2 =$ *"Next president of the U.S.A.?"*

Choices of answers:

O $a_2^1 =$ *"Bush"* `7dkw98`

O $a_2^2 =$ *"Kerry"* `sqp4b3`

O $a_2^3 =$ *"Nader"* `w832ds`

…

Poll question K:

$q_K =$ *"Vote for a party ?"*

Choices of answers:

O $a_K^1 =$ *"Labour"* `t4s6my`

O $a_K^2 =$ *"Conservative"* `3cef8r`

O $a_K^3 =$ *"Liberal"*… `p83hz5`

Figure 5: Ballot paper with codes.

The voter cards with authorization information of the voters are printed independently. Only hashed or encrypted voter information is stored in the electronic voter registry. The generation of the code lists is independent of any information about the voter. Therefore, the code lists are anonymous and can be randomly distributed together with the voter cards. It is important to note that the deletion of the links between codes and answers and the printing of the voting documents are organizational procedures, which

have to be secured, e.g. by a maker checker principle and other mechanism. The initialization phase must be correctly completed before the (electronic) ballot box is opened for the voting or election.

2.2. Voting or election

The voter chooses his preferred voting channel. In electronic voting over insecure channels, the voter assembles her/his vote decision using the anonymous code list that was provided together with the ballot papers. The code of the vote is submitted together with the voter's credentials. Figure 6 shows an example of the m-voting protocol using a mobile phone with SMS. The mobile number should not be transferred to the m-voting server. The network service provider or telecom supplier should not store any communication information.

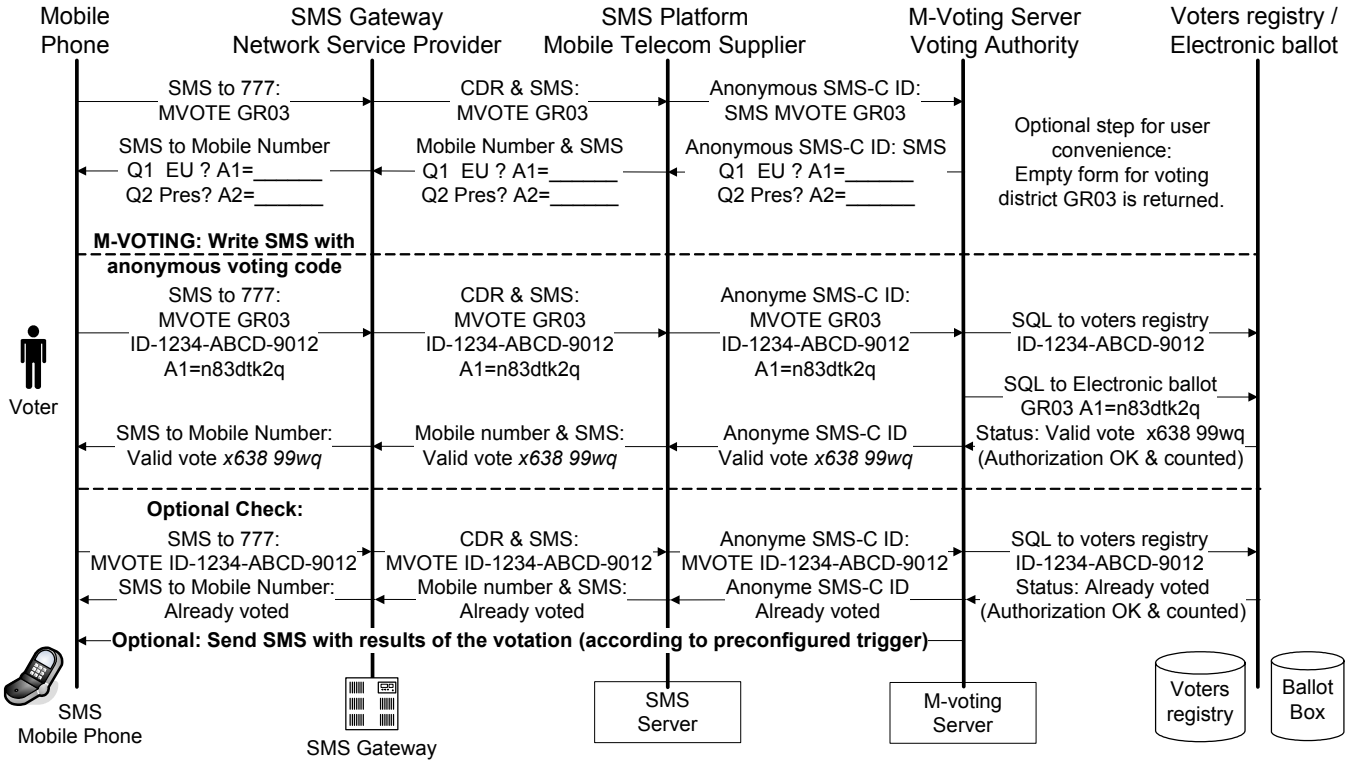

Figure 6: M-Voting via SMS.

On the system side, first the code of the vote is checked for validity by using the control character. If the code is invalid, a notification will be given. Otherwise the authorization of the voter is being checked. The hash of the voter's identification is therefore compared with the hashed voter identifications that were previously stored in the electronic voter registry. Only authorized voter may submit their vote. The electronic voter registry is updated to ensure that a voter may not submit more than one vote – on any channel. In order to assure an auditable system, a central registry with timestamps will be used. The code of the vote is separated from the voter's credentials and cast into the electronic ballot box. The cast votes are mixed randomly and stored in the electronic ballot box.

Electronic ballot box

Codes as stored in the ballot box :

$d_{k,m}$: codes per poll question $k = 1,..., K$ and $m = 1,..., \overline{M}$,

where $\overline{M}$ is the number of cast votes

Ballot box with stored codes for each poll question	**Reassignment** of stored codes to encrypted, anonymous answers from code table	**Decryption** of cast answers with corresponding independent private keys	**De-anonymization** by deletion of random bits from cast answers	**Tallying** of vote per poll question
Electronic ballot box Poll question q_k : $k = 1,...,K$ Stored codes of $\overline{M}$ voters : $d_{k,m}$: m = $1,...,\overline{M}$	Map back codes $\forall k = 1,...,K,\ m = 1,...,\overline{M}$ $d_{k,m} \mapsto e_{k,m}$, where : $d_{k,m} = c \wedge (e_{k,m}, c) \in T$	Decryption f^{-1} $\forall k = 1,...,K,\ m = 1,...,\overline{M}$ $e_{k,m} \mapsto g_{k,m} = f^{-1}(e_{k,m})$ where : $g = f^{-1}(e) = f_1^{-1} \circ f_2^{-1}(e)$ $= \left(e^{\sigma_2} \bmod (n_2)\right)^{\sigma_1} \bmod (n_1)$ with RSA private keys $f_1^{-1} : (n_1, \sigma_1), f_2^{-1} : (n_2, \sigma_2)$	Extract poll answers : $g_{k,m} \mapsto a_{k,m} = h_{k,m}$ where : $g_{k,m} = (i, random)$ $\wedge\ h_{k,m} = h_i$	Result r_k^j of the vote per poll question q_k and choice of answer $a_k^j \in A_k$ $\forall k = 1,...,K, j = 1,...,\lvert A_k \rvert$ $r_k^j = \sum_1^{\overline{M}} \delta(a_{k,m}, a_k^j),$ where $\delta(x, y) = \begin{cases} 1 : x = y \\ 0 : x \neq y \end{cases}$
Example: Poll question 1 : q_1 = *"Joint the EU ?"* Stored codes of $\overline{M}$ voters: $\{d_{1,m}\} = \begin{Bmatrix} d_{1,1},..., \\ \texttt{"n83k2q"}, \\ ..., d_{1,\overline{M}} \end{Bmatrix}$	Example: Code table T $(d_{1,\tilde{m}} \leftrightarrow e_{1,\tilde{m}})$ with $d_{1,\tilde{m}} = \texttt{"n83k2q"} \mapsto$ $e_{1,\tilde{m}} = (\underset{0}{\vert} 04F...739 \underset{\delta}{\vert})$	Example: $\{e_{1,1}...e_{1,\overline{M}}\} \mapsto \{g_{1,1}...g_{1,\overline{M}}\}$ $e_{1,\tilde{m}} = (\underset{0}{\vert} 04F...739 \underset{\delta}{\vert}) \mapsto g_{1,\tilde{m}} = (\underset{0}{\vert} 86C...D25 \underset{\gamma}{\vert})$ $g_{1,\tilde{m}} = (\underset{0}{\vert} 86C...D25 \underset{\gamma}{\vert})$ $= (\underset{0}{\vert} 86C...1F0 \underset{56}{\vert} 0001 \underset{72}{\vert} 2C3...D25 \underset{\gamma}{\vert})$ random i random $\Rightarrow i = 1 \qquad \Rightarrow h_1 = h_{1,\tilde{m}} = "yes" = a_{1,\tilde{m}}$		Example: Poll question 1 : q_1 = *"Join the EU ?"* Poll results : $1,...,\tilde{m},...,\overline{M}$ $r_1^1(a_1^1 = "yes") = ...+1+...$ $r_1^2(a_1^2 = "no") = ...+0+...$ $r_1^3(a_1^3 = "blank") = ...+0+...$

Figure 7: Overview: tallying phase.

2.3. Tallying

When the voting period is finished, the codes of the votes in the ballot box are mapped back to the corresponding encrypted randomized vote answers that are stored in the code table (figure 7). Then the encrypted randomized vote answers are decrypted with the (two or more) private keys of the authorities. If the size of the code table T is smaller than the number of cast votes for all voters ($L|H| < \overline{M}\ K$), then the effort to decipher the whole code table is smaller than the effort to decipher each single vote. The actual vote answers are extracted by removing the random bits.

Decrypt and decipher votes

$\forall k = 1,...,K,\ m = 1,...,\overline{M}$:

Map back the codes to encrypted randomized vote answer

$$d_{k,m} \mapsto e_{k,m} : d_{k,m} = c \wedge (e_{k,m}, c) \in T$$

Decryption f^{-1} with corresponding (two or more) private keys

$$e_{k,m} \mapsto g_{k,m} = f^{-1}(e_{k,m})$$

Extract answers to poll questions from randomized votes by removing the random bits

$$g_{k,m} \mapsto a_{k,m} = h_{k,m}$$

where :

$$g_{k,m} = (i, random) \ \wedge \ a_{k,m} = h_{k,m} = h_i$$

Finally, the number of votes is tallied per choice of answer for each poll question:

Tally the vote per poll question

$\forall k = 1,...,K, j = 1,...,|A_k|$:

Result r_k^j per poll question k $(k = 1,...,K)$ for answer $a_k^j \in A_k, j = 1,...,|A_k|$

$$r_k^j = \sum_{1}^{\bar{M}} \delta(a_{k,m}, a_k^j), \text{ where } \delta(x,y) = \begin{cases} 1 : x = y \\ 0 : x \neq y \end{cases}$$

2.4. Cleanup

The processes in the cleanup phase depend on the respective voting laws and regulations. Usually the anonymous voter registries and the electronic ballots box are archived for a given period of time. Then code table, ballot box and voter registry are deleted, and the voting system is prepared for the next voting.

3. Conclusions

As discussed in this paper, the proposed voting protocol using anonymous code lists can provide the basis for a secure implementation of an e-voting solution over insecure (mobile) channels. We conclude with some underlying design principles and best practices for a secure electronic voting architecture.

- A pragmatic approach focusing on standard situations is recommended. Usually, e-voting is an additional channel, which is not mandatory to use.
- At least same security is required for e-voting as for postal voting.
- The administrative effort of the authorities should be minimized.
- The client user interface and application must be easy and intuitive to use. The m-voting protocol must rely on a minimum amount of short messages.

Acknowledgments

We would like to thank Christoph Balimann, Stefan Fischer, Alexandre Maurer, Manuel Michaud, and Michel Warynski for their useful input, constructive comments, and helpful discussions.

References

e-Voting website of the Canton of Geneva http://www.geneve.ch/evoting/english/ (available in French and partially English, as visited February 15, 2005)

Warynski, Michel. (2003). E-Voting – La Securité dans la perspective des collectivités publiques (French) in: Muralt Müller, Hannah et al. *E-Voting Stämpfli, Berne*, 219-234.

Zimmermann, Frank. (2003). e-Voting – Sicherheitsanspüche aus der Perspektive des privaten Sektors (German), in: Muralt Müller, Hannah et al. *E-Voting Stämpfli, Berne,* 235-254. and references therein.

Center for Scientific Computing Technical Report TR-96-08, May 1996. 1-48 and references therein. Schneier, Bruce. (1996). Applied Cryptography. Second Edition. John Wiley & Sons, 1996.

Hirt, Martin. (2001) Multi-Party Computation: Efficient Protocols, General Adversaries, and Voting. *Reprint as vol. 3 of ETH Series in Information Security and Cryptography. Hartung-Gorre Verlag, Konstanz, 2001,* 1-174 and references therein.

Lüscher, Martin. (1993). A portable high-quality random number generator for lattice field theory calculations, Computer Physics Communications, 79 100-110.

Marsaglia, G. & Zaman, A. (1991). A new class of random number generators. Ann. Appl. Prob.,1(3),

Masuda, Norio & Zimmermann, Frank. (1996). PRNGlib: A Parallel Random Number Generator Library. *Swiss*