Design and Analysis of a Practical E-voting Protocol

Marián Novotný

Institute of Computer Science, Pavol Jozef Šafárik University, Jesenná 5, 041 54 Košice, Slovakia
marian.novotny@upjs.sk

Abstract. In this paper we design an e-voting protocol for an academic voting system which should be independent from other university applications. We briefly discuss security requirements for e-voting schemes focusing on our proposed scheme. We design a receipt-free e-voting protocol which requires neither anonymous channel nor other physical assumptions. We give a short survey on formal analysis of e-voting protocols. Using the applied pi-calculus we model and analyze some security properties of the proposed scheme.

1 Introduction

Voting is one of the most important and fundamental institutions of democratic society. The process of informatisation brings the possibility of cheaper and more comfortable alternative to classical voting – voting through the Internet. The increase of the *turnout* using e-voting is generally disputable [9], but we believe that it can be achieved in the domain of academic field, because of young and more computer proficient participants.

On the other hand it is desirable to protect the *privacy* of voters and shield them from the possibility of *frauds*. Design and analysis of e-voting protocols have become a challenge in cryptography and security research field. Since design of cryptographic protocols is notoriously error-prone, it is necessary to prove security properties using *formal methods*.

There are many schemes [14, 15] which realize different kinds of demands for e-voting. They use various cryptographic primitives [14] such as *blind signature [6], bit commitment, homomorphic encryption, mixnets, zero-knowledge proofs, deniable encryption* [5] etc. We may distinguish three main kinds of protocols in literature according to the mechanism for providing the privacy of voters: blind signature schemes [8, 12], homomorphic encryption schemes [14, 15] and schemes based on mixing the votes [14]. A good survey on e-voting schemes can be found in [14, 15].

This paper is organized as follows. The next section describes the academic voting system and security requirements and phases of e-voting schemes. The section following next introduces the proposed e-voting scheme. In section 4 we give a short survey on formal analysis of e-voting protocols and analyze our

proposed scheme. The last section presents our conclusions and suggestions for the future work.

2 Academic voting system

Nowadays academic institutions are using various applications such as university information system, video-conference, portal of virtual collaboration etc. The evolution of these systems and corresponding demands for them requires to implement modules for providing various private voting services, e. g., *election to academic boards*, *balloting of commissions*, *anonymous questionnaires* about lectures, teachers or *anonymous psychological questionnaires* etc.

Our aim is to design and implement an *academic voting system*, which should be independent from other university applications. These will be extended by a *module for managing of voting*, such as creating and defining voting with obligatory properties as the type of voting, the list of eligible voters and candidates, in the case of questionnaires the questions and possible answers, the start and the deadline for the vote-casting etc. We assume a pre-established *Public Key Infrastructure* with registered conceivable voters with relevant *certificates* of public keys. Each certificate must contain a part, which uniquely identifies various potential voters such as students, teachers, foreign visitors etc.

The act of vote-casting or filling questionnaire will be accomplished by using a *Java Web Start* application in order to have a program without installation which can use a keystore of voter's keys and secrets on his device. The source code of the client-side software will be signed by the trusted registration authority and verified by the user and the *Java Virtual Machine* during loading the application. This way we assume that the client-side will be trusted.

In the beginning of realization of the academic voting system we do not assume *qualified* certificates. It will be sufficient to obtain a certificate by the e-voting client where a user creates a pair – a private key with corresponding public key. By using the client application the user authenticates and authorizes himself in a university application and sends the public key together with the proof of knowledge of the corresponding private key. The university application then issues the certificate of the public key for him.

Security requirements and phases of e-voting schemes. There exist several possible types of voting [14]. According to the requirements for the academic voting system we need to implement *yes/no, 1-L, 1-L-K* and special *"write-in"* [14] for questionnaires.

The stages of voting can be seen on Fig. 1. After creating and defining voting, the process of voting consists of six stages in general. We focus mainly on the phases of *registration*, *vote-casting*, *tallying* and *verification*, which are realized by the e-voting protocol.

A voting scheme is expected to satisfy certain security requirements, which are summarized and compared in [15]. In the following we enumerate and briefly describe these properties which are relevant for the academic voting system.

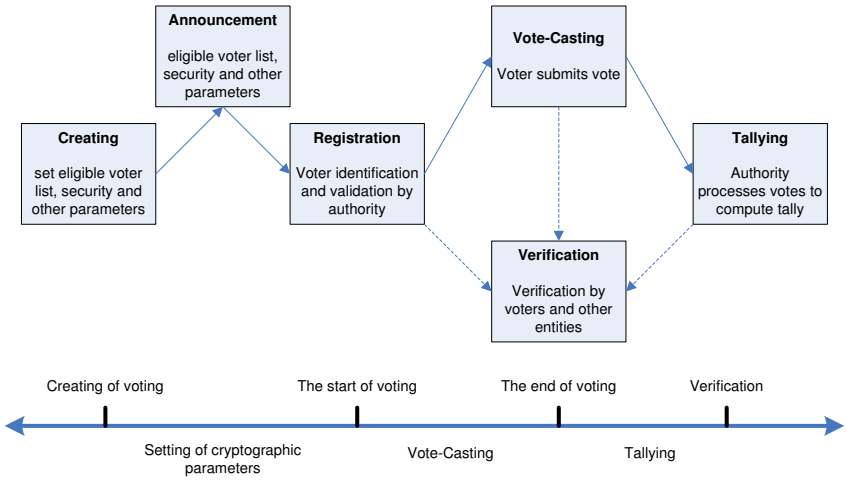

Fig. 1. The stages of a voting scheme

- *Eligibility.* Only valid voters who meet certain pre-determined criteria are eligible to vote.
- *Privacy.* In a secret ballot, a vote must not identify a voter and any traceability between the voter and his vote must be removed.
- *Verifiability.* A voter should be able to verify whether his vote was correctly recorded and accounted in the final vote tally. We distinguish between *individual* and *universal* verifiability. In the latter case not only the voter but anyone can verify that all valid votes were included and the tally process was *accurate*.
- *Dispute-freeness.* A voting scheme must provide a mechanism to resolve all disputes at any stage.
- *Accuracy.* A voting scheme must be error-free. Votes of invalid voters should not be counted in the *final tally*.
- *Fairness.* No one should be able to compute a *partial tally* as the election progresses.
- *Robustness.* A scheme has to be robust against active or passive attacks and faults as well.
- *Receipt-freeness.* A voter should not be able to provide a receipt with which he may be able to prove his vote to any other entity.
- *Practicality.* A voting scheme should not have assumptions and requirements that may be difficult to implement for a real application.

3 The proposed scheme

In the protocol we assume neither anonymous nor other physical assumptions such as *untappable channel* [14]. On the other hand our scheme requires a pre-

established public key infrastructure. In this way each eligible voter has a valid certificate of a public key according to the private key for signing.

We use three servers – two *registration* and one *tally* as sketched in Fig. 2. Each registration server has a list of eligible voters for certain voting. We doubled registration servers, because we consider *blind signature* schemes to be problematic due to the possibility of creating votes of abstain voters by the registration server. The problem can be caused if a voter has been registered and then *abstains*. In this case the single registration server knows registered voters and is able to create a "fake" vote which substitutes a vote of the abstain voter in the final tally. We would like to avoid this problem by using two registration servers for controlling themselves. Machines on which the registration servers will be running should be mounted on different places under the control of different authorities. Moreover they serve in the protocol also as a simple *decryption mixnet* for providing anonymity of the communication in the vote-casting phase.

First we introduce a basic scheme which does not provide the receipt-freeness property. Next we will extend it by using trapdoor bit commitment combined with deniable encryption in order to provide the receipt-freeness property. In the description of the scheme we use the following notations. For a participant X we denote his public key for encryption (signing) Pk_X^E (Pk_X^S) and the corresponding private key Sk_X^E (Sk_X^S). Encryption of a message m under the public key Pk_X^E is denoted as $E_{Pk_X^E}(m)$ and signing the message m by the participant X using his private key Sk_X^S as $S_{Sk_X^S}(m)$.

The scheme based on blind signature. For ensuring the privacy property, i. e., removing any traceability between a voter and his vote we use a blind signature scheme. The concept of the blind signature was introduced by D. Chaum in [6]. This kind of signature solves the problem when a *requester* R wants to sign a message m from an *authority* A without revealing any information about m. The content of the message m for the signer A with the public key Pk_A^S is *blinded* by the requester using the function $Bl(m, r, Pk_A^S)$ with a random parameter r. The signer A signs the blinded message as $S_{Sk_A^S}(Bl(m, r, Pk_A^S))$ and sends it to the requester. The requester retrieves the desired signature using the *unblind* function $Unbl(S_{Sk_A^S}(Bl(m, r, Pk_A^S)), r, Pk_A^S) = S_{Sk_A^S}(m)$.

Registration phase. First a voter V_i obtains public keys of all servers and a public parameter of the tally server for the voting g_T^t. The private parameter t of the tally server can be shared by many authorities such as members of a voting committee and $g_T \in \mathbb{G}$ is a generator of a cyclic group $\mathbb{G}$ on which we can map the set of asymmetric keys for the encryption of a vote. We assume that for the group $\mathbb{G}$ is the *CDH problem* [11] hard. This way we would like to ensure the scheme to be more robust in the sense of the fairness property.

The voter V_i chooses his vote $vote_i$, then he randomly chooses v_i and computes the asymmetric key $K_i = (g_T^t)^{v_i}$ for the decryption with the corresponding key K_i^{-1} for the encryption of the vote. For all public key encryptions including the encryption of the vote we use an *IND-CCA* [11] probabilistic encryption

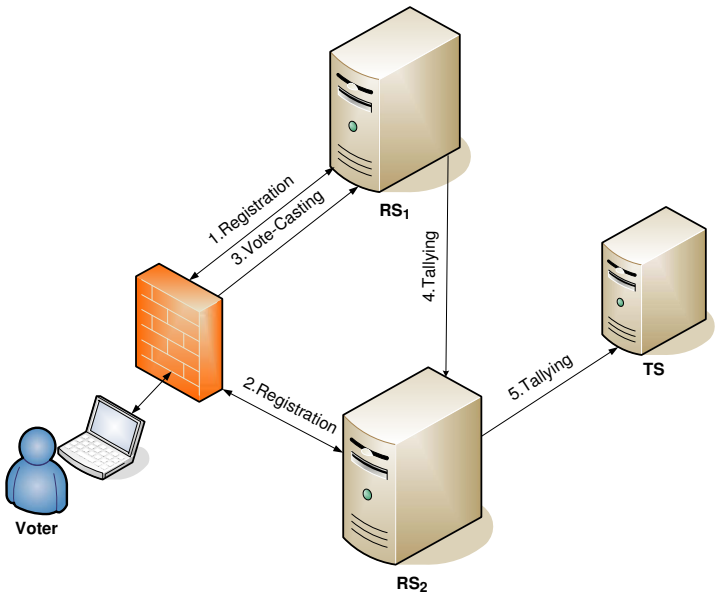

Fig. 2. The servers and the communication in the protocol

scheme and for hashing a hash function H which fulfills appropriate security requirements for hash functions [11].

The voter V_i prepares his *ballot* $b_i = E_{K_i^{-1}}(vote_i), g_T^{v_i}$ and computes its hash value $h_{i_1} = H(b_i)$. Then he blinds this hash value as $bl_{i_1} = Bl_{Pk^S_{RS_1}}(h_{i_1}, r_{i_1})$ by a random parameter r_{i_1}. The voter registers in RS_1 server and signs the hash value of his ballot by RS_1 server in the following messages:

1. $V_i \longrightarrow RS_1\ E_{Pk^E_{RS_1}}(V_i, S_{Sk^S_V}(ID_{voting}, V_i, bl_{i_1}))$
2. $RS_1 \longrightarrow V_i\ S_{Sk^S_{RS_1}}(bl_{i_1})$

After receiving and encrypting the first message the registration server RS_1 examines whether the voter V_i is on the voter's list for the voting ID_{voting} and checks the signature. If the check succeeds, it signs the blinded message bl_{i_1} and sends it back to the voter in the second message. The voter unblinds this message using the random parameter r_{i_1} and obtains the signature of the hash value of his ballot: $S_{Sk^S_{RS_1}}(h_{i_1})$.

The voter V_i after successful registration in RS_2 server will obtain a "token" for the vote-casting phase. First the voter prepares the message $m_{i_2} = E_{Pk^E_{RS_2}}(E_{Pk^E_{TS}}(b_i, S_{Sk^S_{RS_1}}(h_{i_1})))$ and computes its hash value as $h_{i_2} = H(m_{i_2})$. Then he blinds it by a random parameter r_{i_2} thus has $bl_{i_2} = Bl_{Pk^S_{RS_2}}(h_{i_2}, r_{i_2})$. The voter registers in RS_2 server in the following messages:

1. $V_i \longrightarrow RS_2\ E_{Pk^E_{RS_2}}(V_i, S_{Sk^S_V}(ID_{voting}, bl_{i_2}, V_i))$

2. $RS_2 \longrightarrow V_i \; S_{Sk^S_{RS_2}}(bl_{i_2})$

After receiving and encrypting the first message the registration server RS_2 examines whether the voter V_i is on the voter's list for the voting ID_{voting} and checks the signature. If the check succeeds, it signs the blinded message bl_{i_2} and sends it back in the second message. The voter unblinds this message using the random parameter r_{i_2} and obtains $S_{Sk^S_{RS_2}}(h_{i_2})$ which is the "token" for the vote-casting phase.

Vote-casting phase. After the above mentioned registration the voter V_i can abstain or take part in the voting by sending the following message until the deadline of the voting:
$V_i \longrightarrow RS_1 \; (E_{Pk^E_{RS_1}}(m_{i_2}, S_{Sk^S_{RS_2}}(h_{i_2})))$
The registration server RS_1 decrypts the received message, then it examines the validity of the signature of RS_2 server on the hash value of the message m_{i_2} and finally it stores it in its local database together with the signature $S_{Sk^S_{RS_2}}(h_{i_2})$.

Tallying phase. After the deadline of the voting, the registration server RS_1 sends *lexicographically ordered* messages m_{i_2} of all participated voters V_i with corresponding signatures $S_{Sk^S_{RS_2}}(h_{i_2})$ to the registration server RS_2. It also publishes the list of signatures in order to avoid possible disputes. The server RS_2 examines the validity of its signature for each message. If the check succeeds, then it decrypts each message m_{i_2} and sends them lexicographically ordered to the tally server TS with its signature of the hash value of the complete list. Furthermore it publishes the list of signatures of messages which were sent by it.

1. $RS_1 \longrightarrow RS_2 \; m_2, S_{Sk^S_{RS_2}}(h_{i_2})$
2. $RS_2 \longrightarrow TS \; E_{Pk^E_{TS}}(b_i, S_{Sk^S_{RS_1}}(h_{i_1}))$

The tally server TS checks the signature of the RS_2 server on the incoming list and then decrypts each message from the list and checks the signature of the RS_1 server on each ballot b_i. It obtains the shared secret t from shareholders and then for each ballot b_i it computes the key $K_i = (g_T^{v_i})^t$ for the decryption of the vote. After decrypting it publishes t and the list $S_{Sk^S_{RS_1}}(h_{i_1}), (E_{K_i^{-1}}(vote_i), g_T^{v_i}), vote_i$.

3.1 Informal analysis of the protocol

In this part we provide informal arguments about the security properties of the proposed basic scheme. In section 4 we will define a formal model of the protocol and specify and prove some security properties using the applied pi-calculus.

The protocol should provide the fairness property under the assumption that all servers and shareholders of the parameter t do not cooperate in order to know the partial tally. The ballot is encrypted three times under the public keys of all servers. The servers can decrypt the message cooperatively in the vote-casting

phase and obtain the ballot. But for acquiring the vote from the ballot it is required to know the secret parameter t which can be shared by many authorities such as members of voting committee etc.

The privacy property is ensured by the blind signature scheme. During the registration in the RS_1 server, the trace between a voter and his ballot is removed. The voter obtains the "token" for the vote-casting phase during the registration in the RS_2 server. In the vote-casting phase it is possible to deduce the communication link between the sender of the message and the vote by cooperation of all servers. At this stage three serves serve as a small decryption mixnet. The message from the voter in the vote-casting phase is waiting for processing in the RS_1 server until the deadline of the voting. Hiding the communication link can be achieved by a single honest server, which does not cooperate with others.

The voter is authenticated in the registration servers using his signatures of messages during registration. If he registers in the RS_1 server and does not register in the RS_2 server, he is not able to send his vote in the vote-casting phase. If he registers in RS_2 and not in RS_1 and he sends the message in the vote-casting phase, the message is not correct and cannot be counted in the final tally. If the voter has been registered in both servers, he can abstain from voting if he does not send the message in the vote-casting phase. For creating a "fake" vote of an abstain voter it is necessary that two registration servers cooperate. If the voter correctly registers in both servers and sends the message in the vote-casting phase and his vote is not published in the final tally, he can look at the published lists of registration servers and find the problem. If it is necessary he can resend his message with the "token" to the server RS_1.

3.2 A Receipt-free version of the scheme

Instead of the ballot b_i from the above mentioned basic scheme we redefine the ballot and denote it as b_i^{RF} in the receipt-free version scheme in order to send the *bit commitment* of a vote and deniable encryption of the parameter for *opening* the commitment. In this way the tally server can open the bit commitment in one way only, but the voter can fake a coercer about his vote by faking the parameter for opening the commitment. To ensure that the scheme is more robust in the sense of the fairness property it is sufficient to encrypt just the bit commitment under the key K_i^{-1} defined above.

Trapdoor bit commitment. We use the trapdoor bit commitment scheme from the paper [13]. Several parameters p, q, g are generated and published by the voting system, where p, q are primes, $q|p-1$, $g \in \mathbb{Z}_p^\star$ and $q = order(g)$. The voter V_i has own secret α_i and computes $G_i = g^{\alpha_i} \bmod p$. We define the bit commitment $BC(vote_i, r_i) = g^{vote_i} G_i^{r_i} \bmod p$ where $vote_i$ is the vote of the voter V_i and r_i is a random parameter. The voter is able to open the bit commitment as an arbitrary vote $vote^c$[1] by using his secret α_i and from

[1] we denote parameters which depends on a candidate c with the superscript c

the equation $vote_i + \alpha_i r_i = vote^c + \alpha_i r_i^c \bmod q$ he can compute r_i^c such that $BC(vote_i, r_i) = BC(vote^c, r_i^c)$.

We assume that the list of candidates (possible votes) is not very large and the tally server for each candidate c can compute and store g^{vote^c}. In this way it is sufficient for opening to send $BC(vote_i, r_i), r_i, G_i$. The tally server computes $G_i^{r_i} = g^{\alpha . r_i} \bmod p$, then the inverse $(G_i^{r_i})^{-1} \bmod p$. The value of g^{vote_i} computes as $g^{vote_i} = BC(vote_i, r_i).(G_i^{r_i})^{-1} = g^{vote_i}.G_i^{r_i}.(G_i^{r_i})^{-1} \bmod p$ and in order to find the vote $vote_i$ compares g^{vote_i} with pre-computed values of $g^{vote^c} \bmod p$ for each candidate c.

Deniable encryption. We use a public key sender deniable encryption $DE_{Pk_X^{DE}}(m, l)$ of a message m under a public key Pk_X^{DE} with an random parameter l. The public key sender deniable encryption scheme should fulfill the following requirements [5]: only receiver possesses the decryption key and the scheme should be *semantically secure*; with overwhelming probability the value decrypted by the receiver contains no flipped bits; the sender should have an efficient *faking algorithm* ϕ such that for a given ciphertext s, which is encryption of the message m with the random factor l ($s = DE(m, l)$) and a faking message m_f, he can compute $l' = \phi(s, m, l, m_f)$, such that $s = DE(m_f, l')$.

The receipt-free version of the protocol. The voter V_i in the receipt-free version similar to the basic scheme of the protocol randomly chooses v_i and computes the asymmetric key $K_i = (g_T^t)^{v_i}$ for the decryption with corresponding key K_i^{-1} for the encryption of the bit commitment. He prepares the ballot $b_i^{RF} = (E_{K_i^{-1}}(BC(vote_i, r_i)), g_T^{v_i}, G_i, DE_{Pk_{TS}^{DE}}(r_i, l_i))$. For each candidate c he computes r_i^c using his secret α_i and the equation $vote_i + \alpha_i r_i = vote^c + \alpha_i r_i^c \bmod q$. This way the voter can open the bit commitment $BC(vote_i, r_i)$ as the vote of the arbitrary candidate c using computed appropriate value of r_i^c. For each parameter r_i^c the voter is able to compute using the faking algorithm ϕ in the deniable encryption scheme the value $l_i^c = \phi(DE_{Pk_{TS}^{DE}}(r_i, l_i), r_i, l_i, r_i^c)$. In the time of coercion he can show the suitable l_i^c such that $DE_{Pk_{TS}^{DE}}(r_i^c, l_i^c) = DE_{Pk_{TS}^{DE}}(r_i, l_i)$ and $BC(vote_i, r_i) = BC(vote^c, r_i^c)$ for each candidate c. In this way the voter is able to fake the coercer about his vote.

The tally server TS in the tallying phase checks the signature of the RS_1 server on the hash value of the ballot b_i^{RF}. It obtains from shareholders the shared secret t. For each ballot it computes the key $K_i = (g_T^{v_i})^t$ for decryption of $E_{K_i^{-1}}(BC(vote_i, r_i))$ and obtains the bit commitment $BC(vote_i, r_i)$. Then it decrypts deniable encrypted r_i and opens the bit commitment as mentioned above.

On the other hand it is required that the voter or anyone else can verify correctness of the final tally. This way we use the *zero-knowledge proof* from the paper [12]. TS server publishes the list of bit commitments $bc_i = BC(vote_i, r_i)$ with relevant signatures of the RS_1 server. It also publishes the list of votes $vote_i'$ in random order using the *random permutation* π such that $vote_i' = vote_{\pi(i)}$.

More precisely the tally server TS *divides* all votes into disjoint groups so that each group contains at least one candidate if possible. For each group it publishes the list of bit commitments $bc_1, \ldots, bc_k$ and the list of votes $vote'_1, \ldots, vote'_k$. Using the non-interactive version of the zero-knowledge proof [12] it proves that it knows the permutation π and the random parameters r_i for opening the bit commitments, such that $bc_i = BC(vote_i, r_i)$, $vote'_i = vote_{\pi(i)}$ without revealing π, r_i. The description how to calculate the proof can be found in [12].

Unfortunately the known public key sender deniable encryption schemes rapidly *lengthen* a message. For our purposes we need to encrypt the parameter r_i for opening the bit commitment. We can use a simple trick in which we generate random short keys SK_i^c for symmetric encryption of each parameter r_i^c of each candidate c. We use the public key deniable encryption only for the "right" key SK_i for encryption of the parameter r_i which is used for opening the vote $vote_i$. The tally server decrypts deniable encrypted symmetric key SK_i and try to decrypt all encrypted parameters. By using some redundancy for example some bit pattern it can distinguish the "right" parameter r_i and correctly open the bit commitment as $vote_i$. On the other hand, in the time of coercion the voter can show the suitable symmetric key SK_i^c for decrypting $E_{SK_i^c}(r_c)$ and opening the bit commitment as the candidate c.

4 Formal analysis of the proposed scheme

In this section we briefly describe formal modeling of security protocols by the applied pi-calculus and give a short survey on formal analysis of e-voting protocols. Next we will model our proposed scheme and analyze some security properties.

4.1 The applied pi-calculus

The *applied pi-calculus* is a language for describing concurrent processes and their interactions. It is based on the *pi-calculus*, but is intended to be less pure and therefore more convenient to use. The applied pi-calculus is similar to the *spi-calculus* [2]. The key difference between them is in the way of handling of cryptographic primitives. The spi-calculus has a fixed set of cryptographic primitives, while the applied pi-calculus allows us to define less usual primitives by means of an *equational theory* on terms.

We briefly describe the *syntax* and the *operational semantics* of the applied pi-calculus from the paper [1]. *Terms* are defined by means of a *signature* Σ, which is a set of function symbols with arities. The set of terms is built from *names*, *variables* and *function symbols* from Σ applied to other terms. Terms and function symbols are sorted and function symbol application must respect sort and arities. Terms are equipped with an equational theory E, i.e., an equivalence relation on terms that is closed under *substitution* of terms for variables and under application of term *contexts* (terms with a *hole* [1]).

Plain processes are defined as follows. The null process **0** does nothing; $\nu n.P$ generates a fresh name n and then behaves as P; $if\ M = N\ then\ P\ else\ Q$

behaves as P if $E \vdash M = N$ and as Q otherwise; $a(x).P$ receives a message N from channel a and then behaves as $P\{N/x\}$; $\overline{a}\langle N\rangle$.P outputs the message N on channel a and then behaves as P; $P|Q$ executes P and Q in parallel; $!P$ generates an unbounded number of copies of P. Active substitutions generalize the *let* construction. The process $\nu x.(\{N/x\})$ corresponds exactly to *let* $x = N$ *in* P. Moreover we use *let* with the pattern matching of tuples and denote it as *let* $(= x, y) = M$. For successful substitution M must be a tuple and the first part of M must be equal in the equational theory with the value of a variable x.

As for the pi-calculus, the operational semantics of the applied-pi calculus is defined in terms of *structural equivalence* and *internal reduction*. Structural equivalence captures rearrangements of parallel compositions and restrictions and the equational rewriting of the terms in a process. Internal reduction defines the semantics of process synchronization and conditionals. *Observational equivalence* captures the equivalence of processes with respect to their dynamic behavior. Two processes are observational equivalent if no context can distinguish them. The formal definitions of these relations can be found in [1].

4.2 Formal analysis of an e-voting protocol

E-voting protocols use unusual cryptographic primitives such as the blind signature, the trapdoor bit commitment etc. For formal modeling of these protocols it is necessary to model properties of these primitives. In this way the applied pi-calculus allows us to express unusual primitives as equations in the equational theory on terms and therefore is appropriate for modeling this kind of protocols.

Seminal work on analysis of e-voting protocols was done by Delaune, Kremer, Ryan in [10]. Authors of this paper modeled and analyzed FOO-scheme [8] using the applied pi-calculus. They formulated the fairness property as an *reachability* property in the sense that the vote of particular voter is not leaked to an attacker before publishing the final tally. They expressed the eligibility property as an reachability property in the sense that an attacker cannot trick system into accepting his vote. They used the ProVerif tool [4] for an automatic analysis of these properties. Privacy property was expressed as an observational equivalence of two processes which differ in two voters which swapped their votes. This property was proved manually by shoving that two processes are *labeled-bisimilar* [1].

In the paper [7] authors defined the receipt-freeness property as an observational equivalence. Roughly speaking, the protocol following this definition satisfies the receipt-freeness property if there exists a cheater process and the coercer cannot tell the difference between a situation in which the cheater process cooperates with him in order to cast the vote c and one in which the cheater pretend to cooperate with him, but casts the vote a. For defining the *coercion-resistance* property authors of [7] defined the *adaptive simulation* relation. They also showed that in the sense of their definitions the coercion-resistance implies the receipt-freeness and this implies the privacy property. Unfortunately these specifications of security properties cannot be proved automatically by using the ProVerif tool.

In the recent paper [3] Backes et al. presented a general technique for modeling remote e-voting protocols in the applied pi-calculus and automatical verification of their security properties. They formalized three fundamental properties of electronic voting protocols: *inalterability* (votes are not modified), *eligibility* (only eligible voters can vote), and *nonreusability* (every voter can vote only once). This formalization of these properties is by means of correspondence assertions. The main idea is to impose a causality relation among certain protocol events in execution traces. Such formulated properties can be analyzed automatically using ProVerif. The authors also formulated the coercion-resistance and the receipt-freeness properties using observational equivalences. This property can be verified automatically by ProVerif for *biprocesses* [3]. But it still requires non-negligible human effort to transform process specification into biprocesses.

Equational theory of the proposed scheme. In the following we describe the formal model of the receipt-free version of the scheme. The equational theory on terms in the formal model is built from the function symbols and equations from Table 1.

Function	Meaning	Equations
$H/1$	hash function	
$pk/1$	public key according to private key	
$g/0, exp/2$	group exponentiation	$exp(exp(g,a),b) = exp(exp(g,b),a)$
$idvoter/1, getpk/1$	identification of the voter	$getpk(idvoter(k)) = k$
$PE/3, PD/2$	probabilistic encryption (decryption)	$PD(y, PE(pk(y), x, r)) = x$
$DE/3, DD/2$	deniable encryption (decryption)	$DD(y, DE(pk(y), x, r)) = x$
$TBC/3, OTBC/2$	trapdoor bit commitment	$OTBC(TBC(m,r,s),r) = m$
$S/2, checkS/3$	signature and its checking	$checkS(S(m,sk), pk(sk), m) = true$
$getm/2$	getting message from signature	$getm(S(m,sk), pk(sk)) = m$
$bl/3, unbl/3$	blinding (unblinding)	$unbl(pk(sk), S(bl(pk(sk), m, b), sk), b) = S(m, sk)$

Table 1. Function symbols with arities and corresponding equations

Modeling and analysis of the protocol. For the communication between processes we use a public channel c, which is under the complete control of an attacker. We assume that all voters are honest and eligible for voting. The process *voter* uniquely generates his private key, then he computes his *id* which binds his public key and registers this *id* in the process *manager* using the private channel p^m_{voter}. The *manager* sends the *id* of the legitimate voter to the exactly one copy of processes of registration servers RS^S_1, RS^S_2 using their private channels $p^m_{RS_1}, p^m_{RS_2}$. They can obtain from the received *id* the public key of the eligible voter for checking his signature in the registration phase. The *manager* also sends the *id* of the eligible voter to the public channel c. The process *voter* chooses his vote non-deterministically by using the process *vchoser* and the private channel p_{vote}. The possible votes v_a, v_b, v_c are free names which

$$
\begin{array}{rl}
vchooser & \triangleq \overline{p_{vote}}\langle v_a\rangle \mid \overline{p_{vote}}\langle v_b\rangle \mid \overline{p_{vote}}\langle v_c\rangle \\[2ex]
manager & \triangleq p^m_{voter}(id)\ .\ \overline{p^m_{RS_1}}\langle id\rangle\ .\ \overline{p^m_{RS_2}}\langle id\rangle\ .\ \overline{c}\langle id\rangle \\[2ex]
RS_1^S & \triangleq p^m_{RS_1}(id)\ .\ c(m_1)\ .\ \textbf{let}\ (=id, m_2) = PD(Sk^E_{RS_1}, m_1)\ \textbf{in} \\
& \quad \textbf{let}\ Pk_V = getpk(id)\ \textbf{in let}\ (=id, m_3) = getm(m_2, Pk_V)\ \textbf{in} \\
& \quad \textbf{if}\ checkS(m_2, Pk_V, (id, m_3)) = true\ \textbf{then}\ \overline{c}\langle S(m_3, Sk^S_{RS_1})\rangle \\[2ex]
RS_2^S & \triangleq p^m_{RS_2}(id)\ .\ c(m_1)\ .\ \textbf{let}\ (=id, m_2) = PD(Sk^E_{RS_2}, m_1)\ \textbf{in} \\
& \quad \textbf{let}\ Pk_V = getpk(id)\ \textbf{in let}\ (m_3, =id) = getm(m_2, Pk_V)\ \textbf{in} \\
& \quad \textbf{if}\ checkS(m_2, Pk_V, (m_3, id)) = true\ \textbf{then}\ \overline{c}\langle S(m_3, Sk^S_{RS_2})\rangle \\[2ex]
voter & \triangleq \nu Sk^S_V\ .\ \textbf{let}\ id = idvoter(pk(Sk^S_V))\ \textbf{in}\ \overline{p^m_{voter}}\langle id\rangle\ .\ p_{vote}(vote)\ . \\
& \quad \nu v\ .\ \nu\alpha\ .\ \nu r_{tbc}\ .\ \nu r_{DE}\ .\ \nu r_{bl1}\ .\ \nu r_{bl2}\ .\ \nu r_1\ .\ \nu r_2\ .\ \nu r_3\ .\ \nu r_4\ .\ \nu r_5\ .\ \nu r_6\ . \\
& \quad \textbf{let}\ K = exp(g^t, v)\ \textbf{in let}\ g^v = exp(g, v)\ \textbf{in} \\
& \quad \textbf{let}\ b = (PE(pk(K), TBC(vote, r_{tbc}, \alpha), r_1), g^v, DE(Pk^{DE}_{TS}, r_{tbc}, r_{DE}))\ \textbf{in} \\
& \quad \textbf{let}\ h_1 = H(b)\ \textbf{in let}\ bl_1 = bl(Pk^S_{RS_1}, h_1, r_{bl1})\ \textbf{in} \\
& \quad \overline{c}\langle PE(Pk^E_{RS_1}, (id, S((id, bl_1), Sk^S_V)), r_2)\rangle\ .\ c(m_1)\ . \\
& \quad \textbf{let}\ b^S = unbl(Pk^S_{RS_1}, m_1, r_{bl1})\ \textbf{in if}\ checkS(b^S, Pk^S_{RS_1}, h_1) = true\ \textbf{then} \\
& \quad \textbf{let}\ m_2 = PE(Pk^E_{RS_2}, PE(Pk^E_{TS}, (b, b^S), r_3), r_4)\ \textbf{in let}\ h_2 = H(m_2)\ \textbf{in} \\
& \quad \textbf{let}\ bl_2 = bl(Pk^S_{RS_2}, h_2, r_{bl2})\ \textbf{in}\ \overline{c}\langle PE(Pk^E_{RS_2}, (id, S((bl_2, id), Sk^S_V)), r_5)\rangle\ .\ c(m_3)\ . \\
& \quad \textbf{let}\ m_2^S = unbl(Pk^S_{RS_2}, m_3, r_{bl2})\ \textbf{in if}\ checkS(m_2^S, Pk^S_{RS_2}, h_2) = true\ \textbf{then} \\
& \quad \overline{c}\langle PE(Pk^E_{RS_1}, (m_2, m_2^S), r_6)\rangle \\[2ex]
RS_1^M & \triangleq c(m)\ .\ \textbf{let}\ (m_1, m_2) = PD(Sk^E_{RS_1}, m)\ \textbf{in} \\
& \quad \textbf{if}\ checkS(m_2, Pk^S_{RS_2}, H(m_1)) = true\ \textbf{then}\ \overline{c}\langle(m_1, m_2)\rangle \\[2ex]
RS_2^M & \triangleq c((m_1, m_2))\ .\ \textbf{if}\ checkS(m_2, Pk^S_{RS_2}, H(m_1)) = true\ \textbf{then} \\
& \quad \textbf{let}\ m_3 = PD(Sk^E_{RS_2}, m_1)\ \textbf{in}\ \overline{c}\langle m_3\rangle \\[2ex]
TS & \triangleq c(m)\ .\ \textbf{let}\ ((m_1, m_2, m_3), m_4) = PD(Sk^E_{TS}, m)\ \textbf{in} \\
& \quad \textbf{if}\ checkS(m_4, Pk^S_{RS_1}, H((m_1, m_2, m_3)) = true\ \textbf{then let}\ r = DD(Sk^{DE}_{TS}, m_3)\ \textbf{in} \\
& \quad \textbf{let}\ K = exp(m_2, t)\ \textbf{in let}\ bc = PD(K, m_1)\ \textbf{in let}\ vote = OTBC(bc, r)\ \textbf{in} \\[2ex]
Voting & \triangleq \nu Sk^E_{TS}\ .\ \nu Sk^{DE}_{TS}\ .\ \nu Sk^E_{RS_1}\ .\ \nu Sk^S_{RS_1}\ .\ \nu Sk^E_{RS_2}\ .\ \nu Sk^S_{RS_2}\ .\ \nu t\ . \\
& \quad \textbf{let}\ (Pk^E_{TS}, Pk^{DE}_{TS}, Pk^E_{RS_1}, Pk^S_{RS_1}, Pk^E_{RS_2}, Pk^S_{RS_2}, g^t) = \\
& \quad (pk(Sk^E_{TS}), pk(Sk^{DE}_{TS}), pk(Sk^E_{RS_1}), pk(Sk^S_{RS_1}), pk(Sk^E_{RS_2}), pk(Sk^S_{RS_2}), exp(g, t))\ \textbf{in} \\
& \quad \overline{c}\langle(Pk^E_{TS}, Pk^{DE}_{TS}, Pk^E_{RS_1}, Pk^S_{RS_1}, Pk^E_{RS_2}, Pk^S_{RS_2}, g^t)\rangle\ . \\
& \quad !voter \mid !vchooser \mid !manager \mid !RS_1^S \mid !RS_2^S \mid !RS_1^M \mid !RS_2^M \mid !TS
\end{array}
$$

Table 2. The formal model of the protocol in the applied pi-calculus

are known to the attacker. The process *voter* generates all random parameters including v for computing the asymmetric key $K = exp(g^t, v)$ for the decryption of the bit commitment and then it follows the instructions of the scheme as defined in the previous section. Two kinds of processes are running on each registration server: the first for registration (RS_1^S, RS_2^S) and the later for mixing (RS_1^M, RS_2^M). The whole process *voting* consists of creating the private keys of servers and the secret parameter t, publishing corresponding public keys and the public parameter g^t to the public channel c and parallel composition of unbounded copies of all defined processes.

We formulate the eligibly property as an causality relation among protocol events in execution traces of the protocol. We added to the specification of the process TS the event $ENDVOTE(X, Y)$ after accepting of a vote X in a ballot Y. Into the process voter we added the event $BEGINVOTE(X, Y, Z)$ for marking the event of starting of voting of a voter Z, which is intended to vote X in a ballot Y. Using the ProVerif tool we proved reachability of the event $ENDVOTE(X, Y)$ and also we proved the assertions $ENDVOTE(X, Y) \Rightarrow BEGINVOTE(X, Y, Z)$ for unbounded number of copies of processes in the formal model of the protocol from Table 2. This assertion means that for all execution traces it holds that an occurrence of the event $ENDVOTE(X, Y)$ implies that an event in which an honest voter Z begun the vote-casting the vote X in the ballot Y has occurred in the particular trace before.

In order to maintain simplicity we do not distinguish between phases of the protocol. But this possibility is supported by ProVerif tool. In this way we can divide processes into phases and simply formulate the fairness property as an secrecy property of leaking the ballot and the vote before the tallying phase.

5 Conclusions

In our work we designed the concept of the academic voting system, which is independent from university applications. For this system we proposed the receipt-free e-voting scheme which requires neither anonymous channel nor other physical assumptions and is based on the blind signature. This scheme was primarily designed for the academic voting system, but can be implemented for other e-voting applications as well.

In contrast to other blind signature schemes we originally doubled registration servers thus avoiding problems with a corrupted registration server. Moreover we use registration servers in the vote-casting phase for providing anonymity of the communication. This way we improved the FOO-scheme [8] and we do not need to assume an anonymous channel. In the receipt-free version we originally combined the trapdoor bit commitment with deniable encryption. This way we improved the Okamoto scheme [13], which requires an untappable channel for sending the parameter for opening the bit commitment and moreover we save one message for sending it.

For better understanding of the requirements of the protocol we defined the formal model of the scheme using the applied pi-calculus and specified and ana-

lyzed some security properties using ProVerif tool. In the future work we would like to prove privacy-type properties of the proposed scheme. After this analysis we are planning to design and implement the academic voting system which will provide the universal interface for other university applications and enable them to use voting services without the need to implement individual voting systems.

References

1. Abadi, M., Fournet, C.: Mobile values, new names, and secure communication. In Proceedings of the 28th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages. ACM (2001)
2. Abadi, M., Gordon, A. D.: A calculus for cryptographic protocols: the spi calculus. In Proceedings of the 4th ACM Conference on Computer and Communications Security. ACM (1997)
3. Backes, M., Hritcu, C., Maffei, M.: Automated Verification of Remote Electronic Voting Protocols in the Applied Pi-calculus. in Proceedings of 21st IEEE Computer Security Foundations Symposium (2008)
4. Blanchet, B.: An Efficient Cryptographic Protocol Verifier Based on Prolog Rules. In 14th IEEE Computer Security Foundations Workshop. IEEE Computer Society (2001)
5. Canetti, R., Dwork, C., Naor, M., Ostrovsky, R.: Deniable Encryption. In Proceedings of the 17th Annual international Cryptology Conference on Advances in Cryptology. Lecture Notes In Computer Science, vol. 1294. Springer-Verlag (1997)
6. Chaum, D.: Blind signatures for untraceable payments. Advances in Cryptology - Crypto. Springer-Verlag (1983)
7. Delaune, S., Kremer, S., Ryan, M. D.: Coercion-resistance and receipt-freeness in electronic voting. In Proc. 19th IEEE Computer Security Foundations Workshop. IEEE Computer Society Press (2006)
8. Fujioka, A., Okamoto, T., Ohta, K.: A Practical Secret Voting Scheme for Large Scale Elections. In Proceedings of the Workshop on the theory and Application of Cryptographic Techniques: Advances in Cryptology. Lecture Notes In Computer Science, vol. 718. Springer-Verlag (1993)
9. Kersting, N., Baldersheim, H.(Eds.): Electronic Voting and Democracy A Comparative Analysis. Hampshire. Palgrave Macmillan (2004)
10. Kremer, S., Ryan, M. D.: Analysis of an Electronic Voting Protocol in the Applied Pi Calculus. In Proceedings of the European Symposium on Programming. Lecture Notes in Computer Science, vol. 3444. Springer Verlag (2005)
11. Mao, W.: Modern Cryptography: Theory and Practice. Prentice Hall Professional Technical Reference (2003)
12. Okamoto, T.: An electronic voting scheme. IFIP World Conference on IT Tools. Chapman & Hall (1996)
13. Okamoto, T.: Receipt-Free Electronic Voting Schemes for Large Scale Elections. Security Protocols Workshop 1997. Lecture Notes in Computer Science, vol. 1361. Springer Verlag (1998)
14. Rjaskova, Z.: Electronic voting schemes. Master's thesis. Comenius University (2002)
15. Sampigethaya, R., Poovendran, R.: A Framework and Taxonomy for Comparison of Electronic Voting Schemes. Elsevier Computers & Security, vol.25 (2006)