Proceedings of the ACM Conference on Principles on Distributed Computing
(august 2001, Philadelphia, USA).

Practical Multi-Candidate Election System

O. Baudron[1], P.-A. Fouque[1], D. Pointcheval[1], G. Poupard[2], and J. Stern[1]

[1] École Normale Supérieure, Laboratoire d'informatique, 45, rue d'Ulm – F-75230 Paris Cedex 05.
{Olivier.Baudron, Pierre-Alain.Fouque, David.Pointcheval, Jacques.Stern}@ens.fr
[2] SGDN/DCSSI – DCSSI Crypto Lab – 18, rue du Docteur Zamenhof – F-92131
Issy-les-moulineaux. Guillaume.Poupard@m4x.org

Abstract. The aim of electronic voting schemes is to provide a set of protocols that allow voters to cast ballots while a group of authorities collect the votes and output the final tally. In this paper we describe a practical multi-candidate election scheme that guarantees privacy of voters, public verifiability, and robustness against a coalition of malicious authorities. Furthermore, we address the problem of receipt-freeness and incoercibility of voters. Our new scheme is based on the Paillier cryptosystem and on some related zero-knowledge proof techniques. The voting schemes are very practical and can be efficiently implemented in a real system. **Keywords:** Homomorphic cryptosystems, High-Residuosity Assumption, Practical Voting scheme, threshold cryptography

1 Introduction

1.1 Motivation

An electronic voting scheme is a set of protocols which allow voters to cast ballots while a group of authorities collect the votes and output the final tally. We present a practical multi-candidate election system that is scalable, reliable, and can tolerate any number of participants and candidates.

In most of the previous work, the "yes/no" paradigm in which voters can only cast a boolean vote has been used for technical reasons. But this model is not practical since usually one should consider at least the null vote in addition. Moreover, in a practical system, the tally should be computed at different levels in order to give local, regional and national results. Consequently, the election scheme we propose is practically oriented for large groups of voters and multiple candidates with the possibility of partial tally computation. Moreover, previous works have not described the whole election system from the booth to the computation of the tally in intermediary levels such as local, regional and national results. The separation between levels can appear to be a non-cryptographic feature. However, in a practical point of view, this can be used to reduce the storage cost in each local authority and to reduce the computational cost by distributing all the calculations for checking the proofs. Therefore, it is of great importance to address the scalability problem if we want to deploy this system. Moreover the scenario is adapted to numerous practical situations where local results represent also valuable information.

1.2 Related Work

Election schemes [1, 2, 31, 9, 10, 33] were first described by Benaloh [1]. All these voting schemes primarily discuss only "yes/no" vote. Two election models have been proposed so far.

In Benaloh schemes, a voter shares his vote between n authorities so that $(t+1)$-out-of-them can recover it. Next, each authority computes its encrypted share of the tally and finally at least $t+1$ authorities should collaborate to actually compute the tally.

Cramer *et al.* [10] use another model in which all voters send their encrypted votes to a single combiner. Using the homomorphic property of the cryptosystem, this entity computes the encrypted tally in a publicly verifiable way. Then, the combiner forwards it to the authorities and $(t+1)$-out-of-them should recover the tally by running a threshold cryptosystem. This model is optimal for the communications between voters and authorities. Hence, as we use it, we need a threshold cryptosystem.

In this paper, we address election schemes where multiple candidates can be managed. Multi-candidate election schemes have been first investigated by Cramer *et al.* in [9] and further studied in [10]. In this last scheme, the computation of the tally grows exponentially with the number of candidates : $\Omega((\sqrt{\ell})^{p-1})$ where ℓ is the number of voters and p the number of candidates.

The cryptosystem of Paillier [27] provides an efficient decryption algorithm as well as the largest bandwidth among all cryptosystems using a trapdoor to compute the discrete logarithm. Both of these arguments are the main components to design multi-candidate election schemes and finally a threshold version of this cryptosystem appeared in [15] and was rediscovered independently but later in [11].

In this last paper, the authors have a new and original point of view on Paillier scheme and provide another threshold version. They apply this sharing scheme to a multi-candidate voting scheme as we made but with a different cryptographic proof. The complexity of the proof is logarithmic in the number of candidates whereas the complexity of ours is linear. On the other hand, [11] uses complex zero-knowledge proofs of multiplication, and that can potentially make the complexities of the two schemes comparable for a small number of candidates.

Furthermore, their paper is quite different from our paper since we try to build a global system and not only cryptographic primitives. We have a vision of the whole security features of the voting system. Moreover, our solution takes into account anonymity and receipt-freeness properties.

1.3 Achievements

Our voting system is efficient and flexible to also take into account the different hierarchy levels. The new voting scheme guarantees the following requirements: privacy of voters, public verifiability, robustness and receipt-freeness. The *privacy of users* ensures that a vote will be kept secret from any coalition of t authorities where t is a system parameter. The *public verifiability* ensures that any party including outside observers can convince herself that the election is fair and that the published tally is correctly computed from the ballots that were correctly cast. Next, the *robustness* of the scheme ensures that the system can tolerate faulty authorities who try to cheat during the computation of the tally.

Finally, the *receipt-freeness* property ensures that the voter cannot construct a receipt proving the content of his vote in order to avoid coercibility and "vote buying".

1.4 Outline of the paper

In section 2, we present the organization of the election system. Then, we recall the basic cryptographic tools as Paillier cryptosystem, a threshold decryption algorithm for this scheme, and some zero-knowledge proofs related to a variant of the discrete logarithm problem. In section 4, our complete voting scheme and system is described. Finally, we discuss some related issues such as the practical complexity of the scheme, the anonymity of users and a receipt-free version.

2 Election Organization

2.1 Architecture

In this section we present a *large group-oriented system* that can be used for nation-wide elections. In this case, some organizational constraints have to be carefully taken into account.

For example, the following system presents "direct" elections in a real life scenario :

1. The local center deals with certification of the users and verifies that voters can vote only once. The local authorities also verify correctness of the votes. This task can also be checked by anybody else.
2. The local centers send local results to a regional center that collects all local tallies, verifies that local centers sent correct information and computes the regional results.
3. All regional centers send regional results to the national center which computes the final result and checks whether regional centers have correctly performed their tasks.

2.2 Candidates

In the simplest case, the election aims at choosing one winner out of several candidates. The candidates are potentially physical people, "yes/no" decision, or any arbitrary set of propositions among which a choice has to be made. They are further designated by the numbers $\{1, 2, \ldots, p\}$. They may include a "null" element.

2.3 Players

The architecture involves several entities at different levels.

Voter A voter is a registered person who is allowed to express one vote for one candidate. Each ballot has the same weight in the final result.

Local authority In a local area, ballots are collected by a local authority whose goal is to compute the local result and forward it at a regional level.

Regional authority A regional authority receives local results and computes the regional result which is forwarded to the national authority.
National authority At the top level, the national authority collects regional results, and publishes for each candidate the total number of vote.
Trusted Time Stamp This player guarantees that a player has voted before a certain time.

2.4 Communication Model

The communication model we use is a public broadcast channel with memory and can be implemented with a bulletin board [1]. All communication with the bulletin board are public and can be universally monitored. No party can erase any information but each voter can enter his part of the board.

To control the connection between voters and the bulletin board, an access control must be used.

2.5 Security Requirements

Election schemes require the following properties :

Privacy The privacy of users ensures that a vote will be kept secret from any t-coalition of authorities.
Public verifiability It ensures that any party including observers can convince herself that the election is fair and that the published tally is correctly computed from the ballots that were correctly cast.
Robustness The robustness of the scheme ensures that the system can tolerate some faulty authorities who try to cheat during the computation of the tally.
Anonymity The votes cast by voters should be hidden.
Receipt-Freeness The receipt-freeness property ensures that a voter must not be able to construct a receipt proving the content of his vote.

2.6 Other Attacks in Voting System

Our voting system takes into account two different attacks that appear when we want to build a complete system.

Intermediary Authorities Attack This attack involves intermediary authorities who try to falsify the final result of the national authority.
Rushing Attack At the closure time of the voting system, the local authorities reveal their local tally. The system can be protected to withstand a rushing attack of users who try to falsify the tally if they wait the result of the local authority to vote.

3 Cryptographic Tools

3.1 Paillier Cryptosystem

Various cryptosystems based on randomized encryption schemes $E(M)$ which encrypt a message M by raising a basis g to the power M and suitably randomizing this result have been proposed so far [17, 1, 22, 26, 27]. Their security is based on the intractability of various "residuosity" problems. As an important consequence of this encryption technique, those schemes have homomorphic properties.

Homomorphic Properties They can be informally stated as follows:

$$E(M_1 + M_2) = E(M_1) \times E(M_2) \quad \text{and} \quad E(k \times M) = E(M)^k$$

These "algebraic properties" enable to compute with encrypted values without knowing the content of ciphertexts. They are useful when the anonymity of users is required. For example, we can compute the tally without decrypting each vote and therefore can guarantee the privacy of users.

In electronic voting schemes, a variant of the ElGamal encryption scheme [10] has been widely used. Instead of encrypting m with $(g^k \bmod p, my^k \bmod p)$, we compute $(g^k \bmod p, g^m y^k \bmod p)$. Unfortunately, such a scheme cannot be considered as a trapdoor discrete logarithm scheme because no trapdoor exists to determine m given $g^m \bmod p$. Anyway, in "0/1" voting schemes, the cryptosystem only manages small numbers because the number of voters is limited and each voter votes "0" or "1". Consequently, the tally cannot be very large and an exhaustive search allows to give the result.

However, in multi-candidate election schemes the tally can become larger because the encoding of many candidates cannot be reduced and if we want to continue to compute the different results, we have to use an encoding size in $\mathcal{O}(p \times |\ell|)$, where p is the number of candidates and $|\ell|$ is the size of the number of users. For example, a national election may involve 10 candidates and hundred millions voters needs the ability to encrypt 266-bits messages. In such applications, the modified ElGamal scheme can no longer be used since exhaustive search, or more efficient methods like index calculus algorithm, cannot efficiently recover the tally. A solution is to use a trapdoor discrete logarithm scheme with large bandwidth such as Naccache-Stern [22], Okamoto-Uchiyama [26], or Paillier [27].

Description of Paillier scheme Paillier has presented three closely related such cryptosystems in [27]. We only recall the first one.

- **Key Generation.** Let N be an RSA modulus $N = pq$, where p and q are prime integers. Let g be an integer of order a multiple of N modulo N^2. The public key is $PK = (N, g)$ and the secret key is $SK = \lambda(N)$ where $\lambda(N)$ is defined as $\lambda(N) = \mathrm{lcm}\,((p-1)(q-1))$.
- **Encryption.** To encrypt a message $M \in \mathbb{Z}_N$, randomly choose x in $\mathbb{Z}_N^*$ and compute the ciphertext $c = g^M x^N \bmod N^2$.
- **Decryption.** To decrypt c, compute $M = L(c^{\lambda(N)} \bmod N^2)/L(g^{\lambda(N)} \bmod N^2) \bmod N$ where the L-function takes in input elements from the set $\mathcal{S}_N = \{u < N^2 | u = 1 \bmod N\}$ and computes $L(u) = \frac{u-1}{N}$. See in appendix 9.1 for further details.

3.2 Threshold Version of Paillier Cryptosystem

In order to prevent authorities to learn the votes and to protect the privacy of users, we can use a threshold version of Paillier cryptosystem. Therefore, instead of merely decrypt the encrypted tally, each authority uses n servers to share its secret key so that at least t authorities are required to decrypt a vote.

Threshold decryption model The decryption process includes the following players : a combiner, a set of n servers P_i, and users. We consider the following scenario :

- In an initialization phase, the servers use a distributed key generation algorithm to create the public key PK and secret shares SK_i of the private key SK. To remove the trusted dealer, see [12, 14]. Next the servers publish verification keys $\mathrm{VK}, \mathrm{VK}_i$.
- To encrypt a message, any user can run the encryption algorithm using the public key PK.
- To decrypt a ciphertext c, the combiner forwards c to the servers. Using their secret keys SK_i and their verification keys $\mathrm{VK}, \mathrm{VK}_i$, each server runs the decryption algorithm and outputs a partial decryption c_i with a proof of validity of the partial decryption proof_i. Finally, the combiner uses the combining algorithm to recover the cleartext if enough partial decryptions are valid.

This scheme is presented in appendix 9.2.

3.3 Zero-Knowledge Proofs

Given encryptions of Paillier, we review various zero-knowledge proofs that the secret message satisfies some properties. We simply state what can be guaranteed by affixing the proof of the encryption. Details are provided in appendix 9.3.

Proof of knowledge of an encrypted message. When creating the encryption of a message it is possible to prove that one actually knows the encrypted message.

Let N be a k-bit RSA modulus. Given $c = g^m r^N \bmod N^2$, the prover P convinces the verifier V that he knows m similar to Okamoto [25] and Guillou-Quisquater [18].

We note $a \div b$ the quotient in the division of a by b.

1. P chooses at random $x \in \mathbb{Z}_N$ and $s \in \mathbb{Z}_N^*$. He computes $u = g^x s^N \bmod N^2$ and commits to u.
2. V chooses a challenge $e \in [0, A[$ and sends e to P.
3. P computes $v = x - em \bmod N$, $w = sr^{-e} g^{(x-em)\div N} \bmod N$ and sends them to V.
4. V checks that $g^v c^e w^N = u \bmod N^2$.

Proof that an encrypted message lies in a given set of messages. When encrypting a message, it is possible to append a proof that the message lies in a public set $\mathcal{S} = \{m_1, \ldots, m_p\}$ of p messages without revealing any further information.

Proof of equality of plaintexts. When encrypting a message, it is possible to append a proof that two encrypted message are equal.

4 The voting scheme

In this section, we describe the complete protocol of our scheme, using the cryptographic tools presented in section 3. Especially, we wish to emphasis the user's vote generation and the communication between the three hierarchical levels, the national one, the regional one, and the local one.

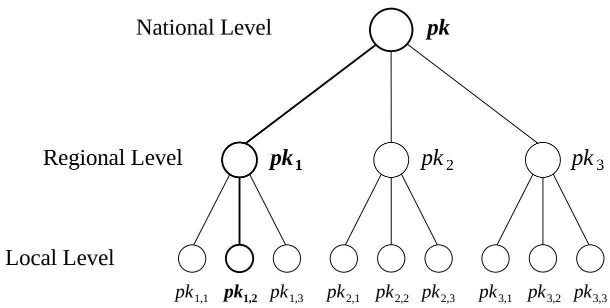

Fig. 1. Organization of the authorities.

4.1 Setup

We consider a 1-out-of-p election where one candidate is chosen among p others. In the initialization process, each authority, at any level generates its public key and certifies it with an independent certification authority. We use the following hierarchical notation: pk for the national authority, pk_i for the regional authorities and $pk_{i,j}$ for the local authorities.

After the preliminary phase, each authority publishes on its own bulletin board the correct public keys: for example, the i^{th} regional authority the keys pk, pk_i, and the j^{th} local authority of the i^{th} regional authority the keys $pk, pk_i, pk_{i,j}$.

We denote by ℓ the number of voters and we set M an integer larger than ℓ. For example, we can choose $M = 2^{\lceil \log_2 \ell \rceil}$, the power of 2 immediately larger than ℓ.

4.2 The Voting Phase

Consider a voter from a local area with public keys $pk, pk_i, pk_{i,j}$ who wishes to vote for the m^{th} candidate. He issues a ballot in the following way:

1. He downloads from the bulletin board the 3 public keys $pk, pk_i, pk_{i,j}$ of his zone.
2. Using each public key, he encrypts the integer M^m with Paillier scheme and generates the ciphertexts C_n, C_r and C_ℓ encrypted respectively with the national, regional and local public key.
3. He generates three proofs to convince any verifier that each ciphertext encrypts a valid vote, i.e. an integer M^m with $m \in \{1, \ldots, p\}$. This is done using the proof that a message lies in a given set.
4. He generates one proof to convince any verifier that the three ciphertexts encrypt the same vote. This is done using proofs of equality of plaintexts. Notice that such a proof is sound because the previous ones guarantees that the size of the encrypted message is bounded.

4.3 Computation of the result

In this subsection we present the computation of the tally. First, we describe the bulletin board of the local authorities which can be accessed by all users for example through the web site of the local authority $A_{i,j}$.

Name	C_ℓ	C_r	C_n
User 1	$g_{i,j}^{v_{i,j,1}}$	$g_i^{v_{i,j,1}}$	$g^{v_{i,j,1}}$
User 2	$g_{i,j}^{v_{i,j,2}}$	$g_i^{v_{i,j,2}}$	$g^{v_{i,j,2}}$
User k	$g_{i,j}^{v_{i,j,k}}$	$g_i^{v_{i,j,k}}$	$g^{v_{i,j,k}}$
User ℓ	$g_{i,j}^{v_{i,j,\ell}}$	$g_i^{v_{i,j,\ell}}$	$g^{v_{i,j,\ell}}$
Sum of $A_{i,j}$	$\sum_k v_{i,j,k}$	$\prod_k g_i^{v_{i,j,k}}$	$\prod_k g^{v_{i,j,k}}$

Fig. 2. Bulletin Board of the local authority.

Figure 2 shows such a bulletin board where the k^{th} user can read and write in his own row but can only read the other rows.

The voters write their name together with their certificates, the three votes and the proofs. Next, they sign all the data in the previous columns of their row and the vote is signed by a time stamp server.

When the voting system is closed, the local authorities have to verify all the proofs and signatures of users and of the time stamp server. Then, the local authorities $A_{i,j}$ compute the product of the correct votes in columns 2 to 4. Next, they act as the combiner in the threshold decryption Paillier cryptosystem for the first product which corresponds to their public key. The decryption can be monitored by outsiders since the decryption process is publicly verifiable.

The local tallies are published in the bulletin board of the regional authorities A_i in column 1 and in column 2 and 3 they write the product of the elements in their own bulletin board. The different authorities $A_{i,j}$ which depend on the authority A_i have the same rights as users in the previous bulletin board. The column C_ℓ is replaced by the decrypted tally. The regional authorities A_i compute the sum in the first column, the products of column 2 and 3, and decrypt

Local Authorities	$V_{j,k}$	$\prod_{j,k} C_r$	$\prod_{j,k} C_n$
Local Auth 1	$\sum_k v_{i,1,k}$	$\prod_k g_i^{v_{i,1,k}}$	$\prod_k g^{v_{i,1,k}}$
Local Auth 2	$\sum_k v_{i,2,k}$	$\prod_k g_i^{v_{i,2,k}}$	$\prod_k g^{v_{i,2,k}}$
Local Auth j	$\sum_k v_{i,j,k}$	$\prod_k g_i^{v_{i,j,k}}$	$\prod_k g^{v_{i,j,k}}$
Local Auth ℓ	$\sum_k v_{i,\ell,k}$	$\prod_k g_i^{v_{i,\ell,k}}$	$\prod_k g^{v_{i,\ell,k}}$
Sum of A_i	$\sum_{j,k} v_{i,j,k}$	$\prod_{j,k} g_i^{v_{i,j,k}}$ $\Downarrow$ decrypts $\sum_{j,k} v_{i,j,k}$	$\prod_{j,k} g^{v_{i,j,k}}$

Fig. 3. Bulletin Board of the regional authority.

the final product in column 2 with the threshold process as the local authorities. At the end, they verify that the tallies computed by the two methods matches. They write on the bulletin board of the national authority their sum in column 1. In column 2, they write the product of their local authorities.

Regional Authorities	$V_{i,j,k}$	$\prod_{i,j,k} C_n$
Regional Auth 1	$\sum_{j,k} v_{1,j,k}$	$\prod_{j,k} g^{v_{1,j,k}}$
Regional Auth 2	$\sum_{j,k} v_{2,j,k}$	$\prod_{j,k} g^{v_{2,j,k}}$
Regional Auth i	$\sum_{j,k} v_{i,j,k}$	$\prod_{j,k} g^{v_{i,j,k}}$
Regional Auth ℓ	$\sum_{j,k} v_{\ell,j,k}$	$\prod_{j,k} g^{v_{\ell,j,k}}$
Sum of A	$\sum_{i,j,k} v_{i,j,k}$	$\prod_{i,j,k} g^{v_{i,j,k}}$ $\Downarrow$ decrypts $\sum_{i,j,k} v_{i,j,k}$

Fig. 4. Bulletin Board of the national authority.

The national authority can compute the product of the elements in column 2 of its bulletin board. Then, it decrypts this product and verifies whether this result is equal to the sum of the elements in column 1.

4.4 Frauds in the computation

The hierarchical subdivisions provide an efficient way of distributing public verifications of the results. Each voter is enable to check whether his vote has been taken into account in the local tally. Then, he recursively performs similar verifications which convinced him that the next level domain correctly included the local tally. Finally, he is guaranteed that his vote is part of the national result. Assuming that the number of hierarchical divisions is d (3 in our discussion) and the total number of voters is ℓ then the computation load is $\mathcal{O}(d^{1/\ell})$ which is easily performed on a single PC.

Additionally, if any error is detected by a given authority the faulty subresults may be discarded from the tally until further recomputation.

5 Enhancements of the scheme

5.1 Time Stamp server

The voting system closes at a precise time T. Hence, the votes of users who have not voted before T will not be taken into account for the computation of the tally. Therefore, the users who have correctly voted before T and whose votes have been refused because of time reasons, must prove that their vote is valid. Consequently, a time stamping system is used to ensure that votes have been performed before T.

5.2 Anonymity

In actual paper-based vote processes, the list of active voters is known, at least by the authority. But it is basically due to the need to control that nobody votes twice.

However, there is no practical reason for having such a list of the active participants and in most situation only the total number of them is required. Moreover, because of the digital format, some large scale analysis can be performed, over many voting processes.

As a consequence, it may be essential to protect anonymity of (non)-voters, while being able to avoid double-voting. As usual, blind signatures [6] are a convenient tool for providing such an anonymity, while preventing double-usage of a certificate.

Let us consider a blind signature scheme that prevents "one-more forgeries" [29]. One can either use the Okamoto-Schnorr [25, 32] version which is based on the difficulty of computing discrete logarithms, or the Okamoto-Guillou-Quisquater [25, 18] version which is based on the difficulty of computing e-th roots modulo a composite number. One may remark that the latter does not add any computational assumption since the RSA problem is stronger than the Higher-Residue problem.

After having proven his identity, each user creates a new pair of matching secret and public keys. Then, with the help of the authority(ies), he gets a certificate on it, using a blind signature process. Of course, the authority have to accept to interact at most once with a user. Thereafter, each voter possesses just one certified public key, which may be used as a pseudonym.

Since we also want to prevent the collusion between a user and the authorities, we have to use a distributed blind signature which requires all the authorities together to help the user to get a valid pseudonym. Therefore, if some authorities would like to help a user to get many pseudonyms to sign more than once, it would be impossible.

An example of such a distributed certification of pseudonyms is given in appendix 9.4.

5.3 Receipt-Free and Incoercible Properties

The concept of "receipt-freeness" was first presented by Benaloh and Tuinstra [2]. Their solution uses a voting booth that physically guarantees two-way secret communication between the authorities and each voter. Next, another receipt-free voting protocol based on a mix-net channel has been proposed by Sako and Kilian [31] where only a one-way secret communication from the authorities to the voters is assumed.

Another class of solutions uses *deniable encryption* [4, 5] such that the voters can lie later how the ciphertext is encrypted and this technique ensures incoercible voters.

We first define both notions of *receipt-freeness* and *incoercibility*. Then, we describe our proposal based on the existence of a secret communication channel between each user and an independant party, called *randomizer*.

Definitions

Receipt-Freeness ensures that the voter cannot provide any receipt proving the content of his vote, even if he wants to.

Incoercibility ensures that a voter cannot be coerced to release his vote.

While the receipt-free property is aimed to prevent users to sell their votes, the incoercible property prevents a coercer to force the voter to reveal his vote. We only focus on receipt-freeness in this article because it is a stronger requirement if we assume that there is no communication between the coercer and the voter during the protocol.

Physical Assumptions To provide receipt-freeness we can use a physical assumption like a tamper-resistant device (such as smartcards) so that the random data used during the voting phase cannot be read by anybody. Therefore, thanks to the semantic security of Paillier's cryptosystem we are sure that anybody cannot learn anything about the vote, even with the help of the voter. However, this assumption may be too strong in some cases, then we will alternatively assume a secret communication channel between any user and a randomizer.

Previous Work The assumption of secret communication channel has also been used in the recent paper of Hirt and Sako [19]. But the communication load of their mix-net to provide receipt-freeness if very high.

Indeed, in their technique, they assume, as we do, that the encryption scheme $\mathcal{E}$ is homomorphic but also allows random re-encryption, which we denote by "$\stackrel{R}{\leftarrow}$".

- each possible vote is first encrypted with a fixed random (say 0)

$$\mathcal{L}_0 = \{\mathcal{E}_{0,1} = \mathcal{E}(1;0), \mathcal{E}_{0,2} = \mathcal{E}(2;0), \ldots, \mathcal{E}_{0,p} = \mathcal{E}(p;0)\}.$$

- each authority permutes and re-encrypts all the votes, with a random permutation, but also new random values so that the voter cannot prove the content of an encryption

$$\mathcal{L}_i = \{\mathcal{E}_{i,1} \stackrel{R}{\leftarrow} \mathcal{E}_{i-1,\pi_i(1)}, \mathcal{E}_{i,2} \stackrel{R}{\leftarrow} \mathcal{E}_{i-1,\pi_i(2)}, \ldots, \\ \mathcal{E}_{i,p} \stackrel{R}{\leftarrow} \mathcal{E}_{i-1,\pi_i(p)}\}$$

- each authority privately communicates the permutation π_i to the user, so that the user can follow the vote he wants to cast
- each authority privately proves that this permutation has really been used, in a zero-knowledge way (without revealing the new random): for each j, $\mathcal{E}_{i,j}$ and $\mathcal{E}_{i-1,\pi_i(j)}$ encrypt the same message
- each authority publicly proves in zero-knowledge that there exists such a permutation: for each j, $\mathcal{E}_{i-1,j}$ has been re-encrypted in $\mathcal{L}_i$
- finally, the user who is the only one able to follow his vote, points it in the final list.

The main drawback is that, on the public bulletin board, the public proofs must appear. In that technique, each authority proves that each message has been re-encrypted in the new list, which makes a very huge amount of data to store.

Independent Randomizers Our proposal is in the same vein as the "self-scrambling anonymizers" [28]. Users ask an external entity to randomize their votes, without modifying the contents. But the voter requires more than just a new encryption of his vote:

- he wants a proof that this new ciphertext encrypts the same vote as his original one. However, that proof must not be transferable, otherwise it would provide a receipt, since he can prove the content of the original encryption. But this proof must also be zero-knowledge in order to leak no information about the new random value used in the encrypted vote. Therefore, either they use an interactive zero-knowledge proof, which is non-transferable, as any zero-knowledge proof, thanks to the simulatability of the transcript. Or they use a non-interactive designated-verifier proof [21].
- he needs a proof that the new encrypted vote is valid. However, without the additional random value introduced by the randomizer, he can no longer provide such a proof by himself. Therefore, he has to interact with the randomizer to get it. However, we want that any part of the transcript of this interaction cannot be used as a receipt by a user. Thus we will use the divertible property [23, 3, 20, 7, 8] of the previous interactive proof that an encrypted message lies in a given set. With such a divertible proof, the transcript seen by the user is independent of the resulting proof: the resulting proof (signature) does not contain any subliminal channel that would enable the user to hide/choose something to make a receipt.
- since three levels (and possibly more) are involved in our mechanism, the randomizer has to randomize the three votes, and the user needs a proof that the three resulting votes encrypt the same message. A divertible version of the proof of equality of plaintexts can also be used. See 9.5 for technical details.

6 Practical Complexity

Let us consider the practical complexity of the voting phase described is section 4. A vote consists of three ciphertexts C_n, C_r and C_ℓ and of some proofs of correctness. Some well known optimizations can be applied. For example, the commitments can be replaced by their hash values as described in [16].

Let us note $|H|$ the size of the hashed commitments, $|A|$ the size of the challenges and $|N|$ the size of the modulus used in the Paillier cryptosystem. The size of a vote is exactly

$$4|H| + 4|A| + (11 + 9p)|N|$$

where p is the number of candidates. Consequently, the communication complexity of the scheme is linear in the size of Paillier modulus, in the number of candidates and in the number of voters. Furthermore, the computation complexity is also $\Theta(p \times |N|)$ modular multiplication for both vote generation and verification.

In practical application, we may choose $|H| = 80$, $|A| = 80$ and $|N| = 1024$. For a 10 candidates election scheme, the size of a vote is about 12.5 KBytes long.

For the complexity of the system, the use of several levels can be used the reduce the computational cost and memory cost. We can see that these costs are logarithmic in the number of levels.

7 Conclusion

In this paper we have described a practical multi-candidates election system which guarantees voting scheme requirements and is scalable so that any number of voters and candidates could be used. Moreover, the system can be adapted to real life scenario with different levels of authorities.

All cryptographic proofs have been optimized to provide an efficient voting scheme.

8 Acknowledgements

The authors are grateful to the anonymous referees for their helpful and detailed comments.

References

1. J. Benaloh. *Verifiable Secret-Ballot Elections.* PhD thesis, Yale University, 1987.
2. J. Benaloh and D. Tuinstra. Receipt-free secret-ballot elections. In *Proc. 26th ACM Symposium on the Theory of Computing (STOC)*, pages 544–553. ACM, 1994.
3. M. Burmester and Y. Desmedt. All Language in NP Have Divertible Zero-Knowledge Proofs and Arguments under Cryptographic Assumptions. In *Eurocrypt '90*, LNCS 473, pages 1–10. Springer-Verlag, 1991.
4. R. Canetti, C. Dwork, M. Naor, and R. Ostrovsky. Deniable Encryption. In *Crypto '97*, LNCS 1294, pages 90–104. Springer-Verlag, 1997.
5. R. Canetti and R. Gennaro. Incoercible Multiparty Computation. In *Proc. 37th IEEE Symposium on the Foundations of Computer Science (FOCS)*. IEEE, 1996.
6. D. Chaum. Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Communications of the ACM*, 24(2):84–88, February 1981.
7. L. Chen. *Witness Hiding Proofs and Applications.* PhD thesis, Aarhus University, August 1994.

8. L. Chen, I. B. Damgård, and T. P. Pedersen. Parallel Divertibility of Proofs of Knowledge. In *Eurocrypt '94*, LNCS 950, pages 140–155. Springer-Verlag, 1995.
9. R. Cramer, Y. Frankel, B. Schoenmakers, and M. Yung. Multi-Authority Secret-Ballot Elections with Linear Work. In *Eurocrypt '96*, LNCS 1070, pages 72–83. Springer-Verlag, 1996.
10. R. Cramer, R. Gennaro, and B. Schoenmakers. A Secure and Optimally Efficient Multi-Authority Election Scheme. In *Eurocrypt '97*, LNCS 1233, pages 113–118. Springer-Verlag, 1997.
11. I. Damgård and M. Jurik. A Generalisation, a Simplification and Some Applications of Paillier's Probabilistic Public-Key System. In *PKC '01*, LNCS 1992, pages 119–136. Springer-Verlag, 2001.
12. I. Damgård and M. Koprowski. Practical Threshold RSA Signatures Without a Trusted Dealer. In *Eurocrypt '01*, LNCS, pages –. Springer-Verlag, 2001.
13. A. Fiat and A. Shamir. How to Prove Yourself: practical solutions of identification and signature problems. In *Crypto '86*, LNCS 263, pages 186–194. Springer-Verlag, 1987.
14. P. Fouque and J. Stern. Fully Distributed Threshold RSA under Standard Assumptions. 2001. in Submission, available on the eprint server `http://eprint.iacr.org`.
15. P. A. Fouque, G. Poupard, and J. Stern. Sharing Decryption in the Context of Voting or Lotteries. In *Financial Crypto '00*, LNCS. Springer-Verlag, 2000.
16. M. Girault and J. Stern. On the Length of Cryptographic Hash-Values used in Identification Schemes. In *Crypto '94*, LNCS 839, pages 202–215. Springer-Verlag, 1994.
17. S. Goldwasser and S. Micali. Probabilistic Encryption. *Journal of Computer and System Sciences*, 28:270–299, 1984.
18. L. C. Guillou and J.-J. Quisquater. A Practical Zero-Knowledge Protocol Fitted to Security Microprocessor Minimizing Both Transmission and Memory. In *Eurocrypt '88*, LNCS 330, pages 123–128. Springer-Verlag, 1988.
19. M. Hirt and K. Sako. Efficient Receipt-Free Voting Based on Homomorphic Encryption. In *Eurocrypt '00*, LNCS 1807. Springer-Verlag, 2000.
20. T. Itoh, K. Sakurai, and H. Shizuya. Any Language in IP Has a Divertible $ZKIP$. In *Asiacrypt '91*, LNCS 739, pages 382–397. Springer-Verlag, 1993.
21. M. Jakobsson, K. Sako, and R. Impagliazzo. Designated Verifier Proofs and Their Applications. In *Eurocrypt '96*, LNCS 1070, pages 143–154. Springer-Verlag, Berlin, 1996.
22. D. Naccache and J. Stern. A New Cryptosystem based on Higher Residues. In *Proc. of the 5th CCS*, pages 59–66. ACM press, 1998.
23. K. Ohta and T. Okamoto. Divertible Zero-Knowledge Interactive Proofs and Commutative Random Self-Reducibility. In *Eurocrypt '89*, LNCS 434, pages 134–149. Springer-Verlag, 1990.
24. K. Ohta and T. Okamoto. On Concrete Security Treatment of Signatures Derived from Identification. In *Crypto '98*, LNCS 1462, pages 354–369. Springer-Verlag, 1998.
25. T. Okamoto. Provably Secure and Practical Identification Schemes and Corresponding Signature Schemes. In *Crypto '92*, LNCS 740, pages 31–53. Springer-Verlag, 1992.
26. T. Okamoto and S. Uchiyama. A New Public Key Cryptosystem as Secure as Factoring. In *Eurocrypt '98*, LNCS 1403, pages 308–318. Springer-Verlag, 1998.
27. P. Paillier. Public-Key Cryptosystems Based on Discrete Logarithms Residues. In *Eurocrypt '99*, LNCS 1592. Springer-Verlag, 1999.
28. D. Pointcheval. Self-Scrambling Anonymizers. In *Financial Crypto '00*, LNCS. Springer-Verlag, 2000.
29. D. Pointcheval and J. Stern. Provably Secure Blind Signature Schemes. In *Asiacrypt '96*, LNCS 1163, pages 252–265. Springer-Verlag, 1996.
30. D. Pointcheval and J. Stern. Security Proofs for Signature Schemes. In *Eurocrypt '96*, LNCS 1070, pages 387–398. Springer-Verlag, 1996.
31. K. Sako and J. Kilian. Receipt-free mix-type voting scheme - A pratical solution to the implementation of a voting booth. In *Eurocrypt '95*, LNCS 921, pages 393–403. Springer-Verlag, 1995.
32. C. P. Schnorr. Efficient Signature Generation by Smart Cards. *Journal of Cryptology*, 4(3):161–174, 1991.
33. B. Schoenmakers. Publicly Verifiable Secret Sharing Scheme and Its Application to Electronic Voting. In *Crypto '99*, LNCS 1666, pages 148–164. Springer-Verlag, 1999.
34. A. Shamir. How to Share a Secret. *Communications of the ACM*, 22:612–613, November 1979.
35. V. Shoup. Practical Threshold Signatures. In *Eurocrypt '00*, LNCS 1807. Springer-Verlag, 2000.

9 Appendix

9.1 Paillier cryptosystem

Correctness of the cryptosystem The integers $c^{\lambda(N)} \bmod N^2$ and $g^{\lambda(N)} \bmod N^2$ are equal to 1 when they are raised to the power N so they are N^{th} roots of unity. Furthermore, such roots are of the form $(1+N)^{\beta} = 1 + \beta N \bmod N^2$. Consequently, the L-function allows to compute such values $\beta \bmod N$ and $L((g^M)^{\lambda(N)} \bmod N^2) = M \times L(g^{\lambda(N)} \bmod N^2) \bmod N$.

Security of the cryptosystem Let us denote HR[N] the problem of deciding N^{th} residuosity, i.e. distinguishing N^{th} residues from non-N^{th} residues. The high residuosity assumption is the generalization of the quadratic residuosity used in the Goldwasser-Micali cryptosystem [17]. In this last cryptosystem, given g a non-quadratic residue modulo N, c is the encryption of $m \in \{0,1\}$ iff $c/g^m \bmod N$ is a quadratic residue which can be decided if we know the factorization of N. In Paillier cryptosystem, given g a non N^{th}-residue modulo N^2, c is the encryption of $m \in \mathbb{Z}_N$ iff $c/g^m \bmod N^2$ is a N^{th}-residue.

The semantic security of Paillier scheme with modulus N is equivalent to HR[N] (see [27] for more details). In the following, we refer to the so-called *Decisional Composite Residuosity Assumption* (DCRA) which assumes that HR[N] is intractable.

9.2 Threshold Version of Paillier Cryptosystem

Description This threshold version appeared in [15] along with a security analysis. We call $\Delta = n!$ where n is the number of servers among which the decryption key is distributed using a Shamir Secret Sharing scheme [34]. This threshold version is closed to the Threshold RSA Signature described by Shoup in [35]. Although the following description uses safe prime RSA moduli ($n = pq$, where p and q are safe primes), we show in [14] how to make the robustness proof without such modulus and how to generate in a distributive manner the special RSA moduli that we need.

Key generation algorithm Choose an integer N, product of two safe primes p and q, such that $p = 2p' + 1$ and $q = 2q' + 1$ and $\gcd(N, \varphi(N)) = 1$. Set $m = p'q'$.
Let β be an element randomly chosen in $\mathbb{Z}_N^*$ and then randomly choose $(a,b) \in \mathbb{Z}_N^* \times \mathbb{Z}_N^*$ and set $g = (1+N)^a \times b^N \bmod N^2$.
The secret key $SK = \beta \times m$ is shared with the Shamir scheme: let $a_0 = \beta m$, randomly choose t values a_i in $\{0, \ldots, N \times m - 1\}$ and set $f(X) = \sum_{i=0}^{t} a_i X^i$.
The share s_i of the i^{th} server P_i is $f(i) \bmod Nm$.
The public key PK consists of g, N and the value $\theta = L(g^{m\beta}) = am\beta \bmod N$.
Let $\mathrm{VK} = v$ be a square that generates the cyclic group of squares in $\mathbb{Z}_{N^2}^*$.
The verification keys VK_i are obtained with the formula $v^{\Delta s_i} \bmod N^2$.

Encryption algorithm To encrypt a message M, randomly pick $x \in \mathbb{Z}_N^*$ and compute $c = g^M x^N \bmod N^2$.

Share decryption algorithm The i^{th} player P_i computes the decryption share $c_i = c^{2\Delta s_i} \bmod N^2$ using his secret share s_i. He makes a proof of correct decryption which assures that $c^{4\Delta} \bmod N^2$ and $v^{\Delta} \bmod N^2$ have been raised to the same power s_i in order to obtain c_i^2 and v_i. This proof is a non-interactive statistically zero-knowledge proof of equality of discrete logarithms in a cyclic group of unknown order mN.

Combining algorithm If less than t decryption shares have valid proofs of correctness the algorithm fails. Otherwise, let S be a set of $t+1$ valid shares and compute the plaintext $M = L\left(\prod_{j\in S} c_j^{2\mu_{0,j}^S} \bmod N^2\right) \times \frac{1}{4\Delta^2\theta} \bmod N$ where $\mu_{0,j}^S = \Delta \times \prod_{j'\in S\setminus\{j\}} \frac{j'}{j'-j} \in \mathbb{Z}$

The Δ factor is used in order to obtain integers and to avoid the computation of inverses modulo the secret value m. Therefore, the Lagrange interpolation formula implies : $\Delta f(i) = \sum_{j\in S} \mu_{i,j}^S f(j) \bmod mN$ and $c^{4\Delta^2 m\beta} = \prod_{j\in S} c_j^{2\mu_{0,j}^S} = c^{4\Delta \sum_{j\in S} \mu_{0,j}^S s_j} \bmod N^2$

9.3 Zero-knowledge Proofs

For convenience, we present only the interactive version of the protocols. Using the Fiat-Shamir paradigm [13], the verifier may be replaced by a hash function to form a non-interactive proof. In the random oracle model, security is guaranteed, provided the interactive scheme is zero-knowledge against a honest verifier [30, 24]. In the following, k is a security parameter. Later on we consider that all parameters are functions of k. In order to simplify the notations, we do not write the dependencies on k but when we say that an expression f is negligible, this means that f depends on k and that, for any polynomial P in k and for large enough k, $f(k) < 1/P(k)$.

Proof of knowledge of an encrypted message. Let N be a k-bit RSA modulus. Given $c = g^m r^N \bmod N^2$, the prover P convinces the verifier V that he knows m similar to Okamoto and Guillou-Quisquater [25].

1. P chooses at random $x \in \mathbb{Z}_N$ and $s \in \mathbb{Z}_N^*$. He computes $u = g^x s^N \bmod N^2$ and commits to u.
2. V chooses a challenge $e \in [0, A[$ and sends e to P.
3. P computes $v = x - em \bmod N$, $w = sr^{-e} g^{(x-em)\div N} \bmod N$ and sends them to V.
4. V checks that $g^v c^e w^N = u \bmod N^2$.

Theorem 1. *For any parameters A and t such that $1/A^t$ is negligible, it holds that t iterations of the protocol is a zero-knowledge proof of knowledge of m (against a honest verifier).*

Proof.
Completeness. Assume P knows m. Following the protocol it holds that $g^v c^e w^N = g^{x-em} g^{me} r^{Ne} s^N r^{-eN} g^{N((x-em)\div N)} = g^x s^N = u \bmod N^2$. The term $(x-em)\div N$

in w corresponds to the quotient of the reduction modulo N in v. Therefore P is accepted with probability 1.

Soundness. Assume that a cheating prover P^* is accepted with non-negligible probability. By a straightforward argument, P^* passes the protocol for at least two different paths and their split can be computed in time linearly bounded by the inverse of P^*'s advantage. It follows that using P^* as an oracle, one may compute in polynomial time (e_1, v_1, w_1) and (e_2, v_2, w_2) such that $g^{v_1} c^{e_1} w_1^N = u = g^{v_2} c^{e_2} w_2^N \bmod N^2$ using an algorithm similar to [32]. Considering the partial logarithms (the discrete logarithm modulo N), it results that $v_1 + me_1 = v_2 + me_2 \bmod N$ and so $m = (v_2 - v_1)/(e_1 - e_2) \bmod N$ which achieves the extraction of m from P^*.

Simulation. Since we assumed a honest verifier, the challenge is independent from the commitment. Therefore the simulator chooses e at random in $[0, A[$ and picks $v \in \mathbb{Z}_N$ and $w \in \mathbb{Z}_N^*$ until $\gcd(w, N) = 1$. Then he computes u to satisfy the verification equation. The simulation runs in linear time with respect to the number t of rounds.

Proof that an encrypted message lies in a given set of messages. Let N be a k-bit RSA modulus, $\mathcal{S} = \{m_1, \ldots, m_p\}$ a public set of p messages, and $c = g^{m_i} r^N \bmod N^2$ an encryption of m_i where i is secret. In the protocol the prover P convinces the verifier V that c encrypts a message in S.

1. P picks at random ρ in $\mathbb{Z}_N^*$. He randomly picks $p-1$ values $\{e_j\}_{j \neq i}$ in $\mathbb{Z}_N$ and $p-1$ values $\{v_j\}_{j \neq i}$ in $\mathbb{Z}_N^*$. Then, he computes $u_i = \rho^N \bmod N^2$ and $\{u_j = v_j^N (g^{m_j}/c)^{e_j} \bmod N^2\}_{j \neq i}$. Finally, he sends $\{u_j\}_{j \in \{1,\ldots,p\}}$ to V.
2. V chooses a random challenge e in $[0, A[$ and sends it to P.
3. P computes $e_i = e - \sum_{j \neq i} e_j \bmod N$ and
   $v_i = \rho r^{e_i} g^{(e - \sum_{j \neq i} e_j) \div N} \bmod N$ and sends $\{v_j, e_j\}_{j \in \{1,\ldots,p\}}$ to V.
   V checks that $e = \sum_j e_j \bmod N$ and that
   $v_j^N = u_j (c/g^{m_j})^{e_j} \bmod N^2$ for each $j \in \{1, \ldots, p\}$.

Theorem 2. *For any non-zero parameters A and t such that $1/A^t$ is negligible it holds that t iterations of the above protocol is a perfect zero-knowledge proof (against a honest verifier) that the decryption of c is a member of $\mathcal{S}$.*

Proof.
Completeness. In the protocol, P has to commit to $u_1, \ldots, u_p$ as if he were proving in parallel that each c/g^{m_j} is a N^{th}-residue. To this end, he uses the malleability of the challenge that enables him to choose in advance $p-1$ values e_j and computes the corresponding fake commitments $u_j = v_j^N (g^{m_j}/c)^{e_j} \bmod N^2$ where the v_j are the final answers picked at random in $\mathbb{Z}_N^*$.

Soundness. Assume that the decryption of c is not a member of $\mathcal{S}$ and that a cheating prover P^* successfully completes an iteration of the protocol. From the final verifying equation and the expression of c it results that for each $j \in \{1, \ldots, p\}$, $v_j^N = u_j (c/g^{m_j})^{e_j} \bmod N^2$. Taking the partial logarithms, it follows that $0 = \log u_j + (m - m_j) e_j$ and since $m \neq m_j \bmod N$, $e_j = \log u_j / (m_j -$

$m) \bmod N$ for each $j \in \{1,\ldots,p\}$. Finally, from the last verifying equation $e = \sum_j e_j \bmod N$, it follows that $e = \sum_j \log u_j/(m_j - m) \bmod N$ which holds with probability at most $1/A$. If the protocol is iterated t times, then standard arguments show that the probability that P^* passes the protocol cannot significantly exceed $1/A^t$ which is a negligible function of k.

Simulation. The simulator randomly chooses $\{e_j\}_{j\in\{1,\ldots,p\}}$ in $\mathbb{Z}_N$. Note that the sum $e = \sum_j e_j \bmod N$ perfectly simulates the challenge given by a honest verifier. Next the simulator picks at random $\{v_j\}_{j\in\{1,\ldots,p\}}$ in $\mathbb{Z}_N^*$ and computes $\{u_j = v_j^N/c^{e_j} \bmod N^2\}_{j\in\{1,\ldots,p\}}$. The sequence
$\{u_j, e, v_j\}_{j\in\{1,\ldots,p\}}$ is a perfect simulation of one round of the protocol. The whole simulation runs in time linear to the number t of rounds.

In [11], the authors have improved this result. Whereas our proof is linear with respect to the number of candidates, the proof of Damgaard and Juric is logarithmic.

Proof of equality of plaintexts. $N_1,\ldots,N_p$ are p k-bit RSA moduli. Given p encryptions $c_j = g_j^m r_j^{N_j} \bmod N_j^2$, under the assumption that the decryptions of the c_j lie in an interval $[0, 2^\ell[$, the prover P convinces the verifier V that the c_j's encrypt the same message m.

1. P picks at random $\rho \in [0, 2^k[$ and $s_j \in \mathbb{Z}_{N_j}^*$ for each j in $\{1,\ldots,p\}$. Then he computes $u_j = g_j^\rho s_j^{N_j} \bmod N_j^2$ and commits to the u_j.
2. V chooses at random a challenge e in $[0, A[$ and sends it to P.
3. P computes $z = \rho + me$ and $v_j = s_j r_j^e \bmod N_j$ and sends z and the v_j for each $j \in \{1,\ldots,p\}$.
   V checks that $z \in [0, 2^k[$ and that $g_j^z v_j^{N_j} = u_j c_j^e \bmod N_j^2$ for each $j \in \{1,\ldots,p\}$.

Theorem 3. *For any non-zero parameters A, t and ℓ such that $1/A^t$ and $2^{\ell-k}A$ are negligible, it holds that t iterations of the previous protocol is a statistical zero-knowledge proof of membership (against a honest verifier) that elements $\{c_1, ..., c_p\}$ encrypt the same ℓ-bit message.*

Proof.
Completeness. For any $j \in \{1, ..., p\}$, it holds that $g_j^z v_j^{N_j} = g_j^{\rho+xe} s_j^{N_j} r_j^{eN_j} = u_j c_j^e \bmod N_j^2$, with probability 1. Furthermore, since $z = \rho + me$, the inequality $z < 2^k$ holds with probability at least $1 - 2^\ell A/2^k$. Thus, a honest prover P successfully completes t iterations of the protocol with probability $(1-2^{\ell-k}A)^t \approx 1 - 2^{\ell-k}At$. From the assumptions, this probability is overwhelming.

Soundness. Assume there exists i_1 and i_2 in $\{1,\ldots,p\}$ such that c_{i_1} encrypts m_1 and c_{i_2} encrypts m_2 with $m_1 \neq m_2$. Then, from the equalities verified by V

$$\begin{cases} g_{i_1}^z v_{i_1}^{N_{i_1}} = u_{i_1} g_{i_1}^{m_1 e} r_{i_1}^{eN_{i_1}} \bmod N_{i_1}^2 \\ g_{i_2}^z v_{i_2}^{N_{i_2}} = u_{i_2} g_{i_2}^{m_2 e} r_{i_2}^{eN_{i_2}} \bmod N_{i_2}^2 \end{cases}$$

Taking the partial logarithms it follows

$$\begin{cases} z = \log_{g_{i_1}} u_{i_1} + em_1 \bmod N_{i_1} \\ z = \log_{g_{i_2}} u_{i_2} + em_2 \bmod N_{i_2} \end{cases}$$

Since $0 \le z < 2^k$, $0 \le em_i < A \times 2^\ell$ ($\ll 2^k$) we get $-2^k < -A \times 2^\ell < z - em_i < 2^k$, there exists ε_1 and ε_2 in $\{0,1\}$ such that the following equalities hold in the integers:

$$\begin{cases} z = \log_{g_{i_1}} u_{i_1} + em_1 - \varepsilon_1 N_1 \\ z = \log_{g_{i_2}} u_{i_2} + em_2 - \varepsilon_2 N_2 \end{cases}$$

Therefore, if $m_1 \neq m_2$, $e = (\log u_{i_1} - \log u_{i_2} - \varepsilon_1 N_{i_1} + \varepsilon_2 N_{i_2})/(m_2 - m_1)$, which occurs with probability at most $4/A$. After t rounds this probability decreases to $(4/A)^t$

Simulation. Following the previous proof, the same simulation works. However, since the simulator uniformly picks z in $[0, 2^k[$ and not in $[me, 2^k + me[$, only a statistical indistinguishability can be achieved.

9.4 Distributed Certification of Pseudonyms

Let us assume that $Z \in \mathbb{Z}_N^*$ is the signature public key of the authority together with a large enough public exponent e, while the distributed secret key are pairs (x_i, y_i), with $x_i \in \mathbb{Z}_N^*$ and $y_i \in [0, e[$ such that $Z = \prod {x_i}^e a^{y_i} \bmod N$. To get a certificate on his pseudonym V, the user interacts with the authorities after having proven his identity:

1. each authority chooses random $u_i \in \mathbb{Z}_N^*$ and $v_i \in [0, e[$, and computes the commitment $w_i = u_i^e a^{v_i} \bmod N$, that he sends to the user.
2. the user chooses random "blinding factors" $\beta \in \mathbb{Z}_N^*$ and $\alpha, \gamma \in [0, e[$, and computes $w = (\prod w_i) a^\alpha \beta^e Z^\gamma \bmod N$
3. the user computes the challenge $\varepsilon = H(w, V)$, $c = \varepsilon + \gamma \bmod e$ and broadcasts it to the authorities
4. each authority computes $r_i = v_i + cy_i \bmod e$, $s_i = (v_i + cy_i) \div e$ as well as $t_i = u_i x_i^c a^{s_i} \bmod N$. They send (r_i, t_i) to the user.
5. the user computes $r = \sum r_i + \alpha \bmod e$, $s = (\sum r_i + \alpha) \div e$, $s' = (c - \varepsilon) \div \bmod e$ and $t = \prod t_i \beta a^s Z^{-s'} \bmod N$

By construction,

$$\begin{aligned} a^{r_i} t_i^e &= a^{r_i + es_i} u_i^e x_i^{ce} = a^{v_i + cy_i} u_i^e x_i^{ce} = a^{v_i} u_i^e (a^{y_i} x_i^e)^c \\ &= w_i Z_i^c \bmod N. \\ a^r t^e &= a^{r+es} Z^{-es'} (\prod t_i^e) \beta^e = a^{\sum r_i + \alpha} Z^{-es'} (\prod t_i^e) \beta^e \\ &= (\prod a^{r_i} t_i^e) a^\alpha \beta^e Z^{-es'} = Z^{c-es'} (\prod w_i) a^\alpha \beta^e \\ &= Z^{c-es'-\gamma} w = Z^{c-c+\varepsilon} w = Z^\varepsilon w \bmod N. \end{aligned}$$

with $\varepsilon = H(w, V)$, where V is the pseudonym of the user. Thanks to the witness-indistinguishability of the Okamoto-Guillou-Quisquater, that blind signature prevents the one-more forgery: no user can get two pseudonyms to vote twice, unless some authorities co-operate. But because of the distribution of the secret key, all the authorities must co-operate to produce any certificate, in particular for a fraudulent one.

Complexity Instead of first proving his identity, the user can simply sign the challenge c that he sends to the authority. Furthermore, instead of involving all the authorities to produce the pseudonym, one can use a threshold (blind) signature scheme [35]. Therefore, $t+1$ authorities would be enough.

9.5 Receipt-Free property with independent randomizers

Let us briefly describe the first stage, in which the randomizer proves the equivalence between both encrypted values. The user has used r to encrypt his vote in $c = g^{m_i} r^N \bmod N^2$, that he sends to the randomizer. Then, the randomizer introduces a new random s to transform the encrypted vote c into $c' = cs^N = g^{m_i}(rs)^N \bmod N^2$.

1. the randomizer chooses at random $x \in \mathbb{Z}_N$ and computes $u = x^N \bmod N^2$ that he sends to the user.
2. the user chooses a challenge $e \in [0, A[$ that he sends to the randomizer.
3. this latter computes $v = xs^{-e} \bmod N^2$ and sends it to the user.
4. finally, the user can check whether $v^N(c'/c)^e = u \bmod N^2$.

Thanks to the zero-knowledge property of this proof, with a fixed-size parameter A, it is non-transferable. But to reduce the number of interactions, one can use a designated-verifier non-interactive proof [21].

The second stage just consists of a divertible variant of the previous proof that an encrypted message lies in a given set. We therefore use the same notations, where the randomizer can be seen as the verifier from the user point of view, but he finally outputs a non-interactive proof that nobody can link with the original one.

1. the randomizer receives the list of $\{u_j\}_{j\in\{1,\ldots,p\}}$ sent by the user. Then he chooses the blinding factors:
   - a random $\beta \in \mathbb{Z}_N$
   - as well as p values $\{\beta_j\}_{j\in\{1,\ldots,p\}}$ in $\mathbb{Z}_N$ such that $\sum \beta_j = \beta \bmod N$
   - and p values $\{\alpha_j\}_{j\in\{1,\ldots,p\}}$ in $\mathbb{Z}_N^*$.

   Thereafter, he computes
   $$u'_j = u_j \alpha_j^N (c/g^{m_j})^{\beta_j} \bmod N^2.$$
   He finally gets $e' = H(u'_1, \ldots, u'_p)$ and sends $e = e' + \beta \bmod N$ to the user.
2. the user sends a list $\{v_j, e_j\}_{j\in\{1,\ldots,p\}}$ to the randomizer that satisfies
   $$e = \sum_j e_j \bmod N \text{ and } v_j^N = u_j (c/g^{m_j})^{e_j} \bmod N^2$$
   for each $j \in \{1, \ldots, p\}$.

3. the randomizer computes $e'_j = e_j - \beta_j \bmod N$ and $v'_j = v_j \alpha_j s^{e'_j} \bmod N^2$. Finally, he sends the list $\{v'_j, e'_j\}_{j\in\{1,\ldots,p\}}$ to the user.

One can check that

$$\begin{aligned} e' &= e - \beta = \sum_j e_j - \sum_j \beta_j = \sum_j (e_j - \beta_j) \\ &= \sum_j e'_j \bmod N, \\ {v'_j}^N &= v_j^N \alpha_j^N s^{Ne'_j} = u_j (c/g^{m_j})^{e_j} \alpha_j^N s^{Ne'_j} \\ &= u_j (cs^N/g^{m_j})^{e'_j} (c/g^{m_j})^{\beta_j} \alpha_j^N \\ &= u'_j (cs^N/g^{m_j})^{e'_j} = u'_j (c'/g^{m_j})^{e'_j} \bmod N^2 \end{aligned}$$

for each $j \in \{1,\ldots,p\}$, where $e' = H(u'_1,\ldots,u'_p)$. Therefore, the list $\{u'_j, v'_j, e'_j\}_{j\in\{1,\ldots,p\}}$ is a non-interactive proof of the validity of the encrypted vote c'.