MULTIPARTY BALLOT COUNTING IN AN INTERNET VOTING SCHEME

Md. Abdul BASED, Stig Fr. MJØLSNES, Marius Brekke STENBEK
Department of Telematics, Faculty of Information Technology, Mathematics and Electrical Engineering, Norwegian University of Science and Technology (NTNU), O.S. Bragstads Plass 2A, N-7491 Trondheim, Norway, tel.: +47 73 59 43 24,
e-mail:{based,sfm}@item.ntnu.no, stenbek@stud.ntnu.no

ABSTRACT

We present a voting scheme in which shares of the ballots are sent to multiple talliers. The talliers operate in parallel and cooperate together to count the ballots by using the technique of the Secure Multiparty Computations (SMPC) in the Virtual Ideal Functionality Framework (VIFF). The counting process is fair since no single tallier can gain knowledge about the ballots or can count the ballots without cooperating with other talliers.

Keywords: *Multiparty Computations, Internet Voting, Paillier Cryptosystem*

1. INTRODUCTION

There are fundamentally different approaches to voting known in literature. One approach is blind signatures and anonymous channels [2]. The anonymous channels can be implemented using mix-nets or based on some physical assumption, but anonymous channels are quite difficult to implement in practice. In another approach, there are several servers to count the ballots. In this case the servers have the ballots of the voters as secret shares among them [1,3]. The servers cooperate to count the ballots and publish the tally. This papers deals with this approach. The third approach is the use of homomorphic encryption. In this case a voter simply publishes an encryption of his vote. The encryptions can be combined into an encryption of the result, and finally a number of decryption servers can cooperate to decrypt the result [4]. In this approach, the private key is needed to be secret-shared among the decryption servers.

Internet voting allows voters to cast their ballots from any location (remote Internet voting) or from a controlled environment (polling booth based Internet voting). Both of these approaches have benefits and limitations. If we allow a voter to vote from any location then we cannot guarantee that the voter is safe from vote-buyer or coercer. To protect the voter from these corrupted parties, polling booth based Internet voting is preferred. Since this environment is controlled we can expect that the voter is free from the corrupted parties. The voting scheme presented in this paper does not distinguish between remote Internet voting and polling booth based Internet voting.

To protect against vote-buying the voters should not be allowed to produce a receipt to prove which candidate the ballot is cast. We propose a Voter Computer (VC) as a ballot-generator for performing cryptographic tasks on behalf of the voter. The voter selects his vote and sends the vote to the VC. The VC then creates a ballot from the vote and appends a nonce value to the ballot. The VC also sends this nonce value to the voter. The voter uses this nonce value for verifying the counting process.

In our voting scheme, we assume that only eligible voters will be equipped with smartcard containing some private information. The system enforces that only authorized voters can vote. The system uses Secure Multiparty Computations (SMPC) for tallying ballots. The SMPC is a secure computation where a number of parties $P_1,\ldots,P_n$ have *private* inputs $x_1,\ldots,x_n$ by which they compute some function f on these inputs, where $f(x_1,\ldots,x_n)=(y_1,\ldots,y_n)$ such that P_i learns y_i but nothing else. The parties are mutually mistrusting so no input is shared between them. In MPC, the tallying is correct as long as $2/3$ of the parties are honest (for active adversaries) [5]. For passive adversaries, the number is $1/2$.

The structure of this article. Section 2 discusses this work in relation to previously published papers. The voting scheme is presented in Section 3. The Paillier cryptosystem is briefly presented in Section 4, while key generation and ballot construction are described in Sections 5 and 6 respectively. The ballot counting is described in Section 7. We also briefly explain the implementation issues in this section. Section 8 includes the summary of the work and future directions.

2. RELATED WORK

The Norwegian Ministry of Local Government and Regional Development (KRD) decided to start an e-voting pilot project (E-valg 2011 Project) for the municipal and regional elections of 2011. If this pilot project gives a successful result, the project will be continued and extended to be used in the general elections of 2013. The KRD, and the E-valg 2011 project group, has created a website [7] where the general information about the project is available. The KRD has decided that the E-valg 2011 project will be implemented as an open source project with a hope to increase public trust in the voting system [8].

Helios is a verifiable online election [9] system. In this system, when a voter casts a vote, the voter gets a smart tracker to track his vote all the way to the tally. But, no one knows how the voter voted. Only the voter can track his vote and everyone can check the tally. The technology provided by Helios is open.

Secret sharing is an essential part of multiparty computations. The process of distributing a secret among a group of participants by allocating shares of the secret is first published by Shamir [15] in 1979. Blakley also publishes secret sharing in [16] in 1979. The multiparty computation based on Shamir's secret sharing scheme using a multiplication

protocol is presented in [5]. This gives a versatile system where the counting algorithm can be tailored to different election rules. Paillier's Probabilistic Encryption scheme is presented in [14].

A non-interactive zero-knowledge protocol for ballot verification is published in [11] and the universal composability of that protocol is published in [13]. Multiple counting servers are proposed to count and publish the ballots in both [11] and [13]. In [12], the authors present an Internet voting scheme using multiparty computations. Like [11] and [13], multiple multiparty talliers are proposed to count ballots in [12].

Our work is based on the ballot counting described in [11]. We modify the ballot construction part and then implement the ballot counting part using the multiparty compuation in the VIFF platform [10]. Encryption is carried out according to Paillier's Probabilistic Encryption scheme [14].

3. THE VOTING SCHEME

The voting scheme is shown in Fig. 1. The voter authentication is done by using a smartcard with fingerprint technology. The smartcard contains the fingerprint reference-features of the voter, the private key of the voter, the private key for a voter group signature, and the public key of the Authentication Server (AS). The public key of the MPC talliers are fetched from the database using the voter side web application. We assume that the group in the group signature consists of only eligible voters.

First, the voter inserts the smartcard into the smartcard reader and supplies his fingerprint. The smartcard verifies the fingerprint and the web server verifies the smartcard. If the verification is ok, then the voter gets access to the voting web page.

Then the voter chooses a candidate and casts his vote. The voter sends his vote to the Voter Computer (VC). The VC generates a ballot from the vote and adds a randomly generated nonce value with the ballot. We assume that the length of the nonce is long enough to avoid duplicate nonce values. The VC sends this nonce value to the voter. Voter's smartcard signs and encrypts the ballot first using the private key for group signature and the public key of the MPC Tallier (MPCT), then the private key of the voter and the public key of the Authentication Server (AS). The VC sends this signed and encrypted ballots to the AS. The AS decrypts the ballot and verifies the signature of the voter. The MPC talliers fetch the ballots from the AS when voting period is over. The MPC talliers remain idle during the voting period. These talliers verify the group signature, and can also verify the validity of the ballots by using the non-interactive zero knowledge protocol [11]. Then the MPC talliers cooperate to add the ballots and to publish the tally. The MPC talliers also publish the nonce values that were added by the VC. Now, the voters can see the nonce values and verify that the counting process is correct. Since the same nonce values is published by all of the MPC talliers, no single MPC tallier can alter or delete a nonce value without detection.

4. PAILLIER CRYPTOSYSTEM

In this section we briefly describe the Paillier Cryptosystem with application to Internet voting.

Pascal Paillier proposed a cryptosystem in [14] which is a new homomorphic cryptosystem. Paillier cryptosystem is a probabilistic encryption based on computations over the group $\mathbb{Z}_{n^2}^*$, where n is an RSA modulus. This cryptosystem has some very attractive properties. For example, it is homomorphic, and allows encryption of many bits in one operation with a constant expansion factor, and allows efficient decryption. Thus it becomes interesting for many cryptological protocols such as electronic voting and mix-nets.

Though Paillier pointed out that his encryption scheme is homomorphic, and is applicable to electronic voting, in order to apply it in voting, two important building blocks are missing [6]. These are an efficient proof of validity of the ballot and an efficient threshold variant of the scheme so that the tally can be decrypted without allowing a single tallier the possibility of learning how a voter voted.

In our voting scheme, we propose that the MPC talliers prove the validity of the ballots non-interactively using the protocol presented in [12]. Also, the MPC talliers count the ballots using the secure multiparty computation operation.

5. KEY GENERATION

In our voting scheme, key generation is carried out according to Paillier's Probabilistic Encryption scheme. In advance, two values are agreed on by the Voter Computer (VC) and the MPC talliers. These values are: a prime number e which is larger than the total number of voters, and a security parameter k. The MPC talliers now compute $n = p * q$ by picking two primes p and q, both with length $k/2$ bits, such that e divides $p-1$ but does not divide $q-1$.

The MPC talliers' private key is computed as $d = lcm(p-1, q-1)$, where the function $lcm(a,b)$ gives the least common multiple of a and b.

Note that any value λ such that $\lambda \bmod n \in \mathbb{Z}_n^*$ and $\lambda = 0 \bmod d$ may be used as private key in this scheme. Then the MPC talliers randomly pick a base $g < n^2$ such that n divides the order of g. This condition is satisfied if $gcd(L(g^d \bmod n^2), n) = 1$, where the function L is defined as

$$L(x) = \frac{x-1}{n}\ ,\ \ \forall x \in \{x < n^2 | x = 1 \bmod n\}.$$

The key pair generation is now completed, with public key (g,n) and private key d.

6. BALLOT CONSTRUCTION

This section describes how the ballots are created and inserted into a database. Note that the MPC talliers are not involved with ballot generation.

After voter authentication, the voter side web application fetches the public keys $(g_0,n_0),(g_1,n_1),(g_2,n_2)$ from the database. A vote is represented by $V = (v_0, v_1, v_2)$ where $v_i = 1$ if the vote was given for candidate i, and 0 otherwise. For instance, a vote for candidate 2 gives

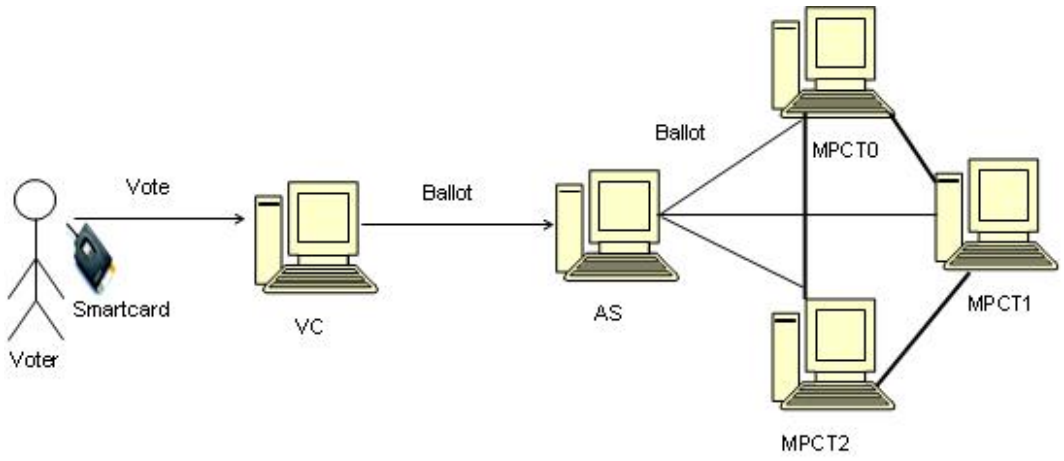

Fig. 1 Internet Voting exemplified with 3 MPC Talliers (MPCT). The Voter's Computer (VC) constructs and sends the ballot to the Authentification Server (AS).

$V = (0,0,1)$. Here, we describe the ballot generation procedure for 3 candidates. This procedure can similarly be extended to an arbitrary number of MPC talliers.

For 3 candidates, nine values $r_{00}, r_{01}, ..., r_{12}, r_{22}$ (where each $r_{ij} \in \mathbb{Z}^*_{n_j}$) are chosen randomly by the Voter Computer (VC). These random values satisfy

$$\begin{aligned} v_0 &= (r_{00} + r_{01} + r_{02}) \bmod e \\ v_1 &= (r_{10} + r_{11} + r_{12}) \bmod e \\ v_2 &= (r_{20} + r_{21} + r_{22}) \bmod e. \end{aligned}$$

Now, the ballot $B = (b_0, b_1, b_2)$ for 3 candidates is generated as follows.

$$b_0 = ({g_0}^{r_{00}} (\bmod\, n_0^2),\ {g_1}^{r_{01}} (\bmod\, n_1^2),\ {g_2}^{r_{02}} (\bmod\, n_2^2))$$

$$b_1 = ({g_0}^{r_{10}} (\bmod\, n_0^2),\ {g_1}^{r_{11}} (\bmod\, n_1^2),\ {g_2}^{r_{12}} (\bmod\, n_2^2))$$

$$b_2 = ({g_0}^{r_{20}} (\bmod\, n_0^2),\ {g_1}^{r_{21}} (\bmod\, n_1^2),\ {g_2}^{r_{22}} (\bmod\, n_2^2))$$

Here, B basically consists of nine cipher texts. That is, the nine r_{ij} plaintexts are encrypted using the MPC talliers' public keys.

The ballot is now ready to be inserted into the database.

Our ballot encryption is modified Paillier encryption [14] in order to easy implementation and make our counting process verifiable.

7. BALLOT COUNTING

7.1. Tally calculation

As mentioned earlier, during the voting period, the MPC talliers remain idle. When all k ballots (k is the total number of ballots cast by the voters) are registered in the database, of AS, each of the MPC talliers is instructed by the authentication server to fetch the ballots and begin the counting procedure.

The first step in this procedure is to compute a tally $S = (s_0, s_1, s_2)$ by multiplying the individual encrypted shares of the k ballots as follows:

$$s_0 = (\prod_i^k b_0^{(i)}[0] \bmod n_0^2\,, \prod_i^k b_0^{(i)}[1] \bmod n_1^2\,, \prod_i^k b_0^{(i)}[2] \bmod n_2^2\,)$$

$$s_1 = (\prod_i^k b_1^{(i)}[0] \bmod n_0^2\,, \prod_i^k b_1^{(i)}[1] \bmod n_1^2\,, \prod_i^k b_1^{(i)}[2] \bmod n_2^2\,)$$

$$s_2 = (\prod_i^k b_2^{(i)}[0] \bmod n_0^2\,, \prod_i^k b_2^{(i)}[1] \bmod n_1^2\,, \prod_i^k b_2^{(i)}[2] \bmod n_2^2\,)$$

Here, $b_x^{(i)}[y]$ is from ballot number i, where $y \leqslant 2$.

The results (s_0, s_1, s_2) of the above multiplications are called sub-tallies. The sub-tallies are decrypted by the MPC talliers, and also used for verifying the counting procedure as explained in the following.

7.2. Sub-tally decryption

As shown in the previous sub-section, each of the nine sub-tallies is a product of ciphertexts encrypted with the MPC tallier's public key. Because of homomorphic properties of the scheme, decrypting all the ballots and computing the sum of the plaintexts is equivalent to decrypting the product of the ciphertexts, i.e. the sub-tallies.

This implies,

$$\sum_i^k D(b_x^{(i)}[y]) = D(s_x[y]).$$

Here, the function $D(c)$ is the decryption of ciphertext c. Homomorphism is achieved because, clearly

$$s_x[y] = \prod_i^k b_x^{(i)}[y] \bmod n_y^2 = \prod_i^k g_y^{r_{xy}^{(i)}} \bmod n_y^2 = g_y^{\sum_i^k r_{xy}^{(i)}} \bmod n_y^2$$

As a result, only the sub-tallies are needed to be decrypted, meaning that the number of ballots does not affect the decryption time complexity. MPC tallier y decrypts $s_x[y]$, where $x \in \{0,1,2\}$. Here,

$$D(s_x[y]) = \frac{L(s_x[y]^{d_y} \bmod n_y^2)}{L(g_y^{d_y} \bmod n_y^2)}.$$

7.3. Ballot counting

When every MPC tallier decrypts its corresponding sub-tally, no more decryption is required. To compute the total tally, the MPC talliers need to collaborate using their sub-tallies. There are two approaches for obtaining this.

In the first approach, each MPC tallier publishes its decrypted sub-tally such that all MPC talliers can calculate the final tally independently.

In the second approach, every MPC tallier inputs its decrypted sub-tally to an agreed Secure Multiparty Computation (SMPC) function which outputs the final tally.

With the first approach the MPC talliers can verify the counting procedure as described in the following.

Publication of the decrypted sub-tallies. With this approach, the MPC talliers publish their decrypted sub-tallies to each other. A MPC tallier may compute the final tally $v = (v_1, v_2, v_3)$, where v_x is the total number of ballots for candidate x.

This is easily calculated as

$$v_x = D(s_x'[0]) + D(s_x'[1]) + D(s_x'[2]).$$

Where $D(s_x'[y])$ is a decrypted sub tally published by tallier y. Note that this gives

$$v_x = \sum_i^k r_{x0}^{(i)} + \sum_i^k r_{x1}^{(i)} + \sum_i^k r_{x2}^{(i)} = \sum_i^k v_x^{(i)},$$

where $r^{(i)}$ is from ballot number i, if all talliers operated as uncorrupted.

Verification of the counting process. When the decrypted sub-tallies are published by the MPC talliers, then one MPC tallier can verify that the other MPC talliers have published the correct sub-tally.

This is possible since the MPC talliers know both the public keys and the plaintexts. So they can encrypt the published plaintexts, and verify that the ciphertexts equal the sub-tallies. More specifically, each MPC tallier checks that for all $x, y = 0,1,2$, the equality

$$g_y^{D(s_x'[y])} \bmod n_y = s_x[y]$$

is correct. If one or more does not hold, the ballot counting should be assumed corrupted.

Secure multiparty computation. It may be desirable that each MPC tallier keeps its decrypted sub tallies secret. But how may the final tally be calculated if the talliers do not publish their individual sub-tallies?

The answer is Secure Multiparty Computation (SMPC). By employing a summation SMPC function, the MPC talliers can find the sum of their decrypted sub-tallies without revealing the actual plaintexts to each other.

Here, we describe how the MPC talliers compute the total number of ballots for candidate 0 using the secure multiparty computation. The MPC talliers repeat the same procedure for candidates 1 and 2. The procedure employs Shamir's secret sharing [15] and secure multiparty addition.

First each server $y \in \{0,1,2\}$ generates a polynomial

$$f_y(x) = p_y + \alpha_y x + \beta_y x^2.$$

Here p_y is MPC tallier y's decrypted sub-tally for candidate 0 ($D(s_0[y])$), while α_y and β_y are secret, random integers chosen by MPC tallier y.

The tallier y now uses the polynomial to generate *secret shares* of p_y:

$$\begin{aligned} f_y(1) &= p_y + \alpha_y + \beta_y \\ f_y(2) &= p_y + 2\alpha_y + 4\beta_y \\ f_y(3) &= p_y + 3\alpha_y + 9\beta_y. \end{aligned}$$

These shares are securely distributed among the MPC talliers such that tallier y will know $f_0(y+1)$, $f_1(y+1)$ and $f_2(y+1)$.

Now, define the function F as $F(x) = f_0(x) + f_1(x) + f_2(x)$. Note that

$$F(0) = p_0 + p_1 + p_2 = \sum_i^k r_{00}^{(i)} + \sum_i^k r_{01}^{(i)} + \sum_i^k r_{02}^{(i)} = \sum_i^k v_x^{(i)},$$

which is the total number of ballots for candidate 0.

This is what the MPC talliers need to compute. Because of how the shares were distributed, each tallier y can now calculate $F(y+1)$.

By publishing these F function values to each other, all talliers may compute $F(0)$ using polynomial interpolation.

Now all of the MPC talliers know the total number of ballots for candidate 0 without revealing their decrypted sub tallies. As mentioned, the MPC talliers need to repeat this procedure twice more to find the number of total ballots given for candidate 1 and 2.

Implementation issues. This functionality is implemented using the Python framework Virtual Ideal Functionality Framework (VIFF) [10]. VIFF is implemented in Python using Twisted and can run on any platform where Python runs in Linux,Windows, or Mac OS X. We like to mention that VIFF is Free Software, licensed under the GNU LGPL.

VIFF allows to do secure multi-party computations, in which for example, three MPC talliers cooperate to count the ballots. Using VIFF, the talliers can count the ballots without revealing anything about their sub-tallies.

The described counting process is implemented in Python, using the Virtual Ideal Functionality Framework (VIFF) for SMPC. 'Virtual Ideal Functionality' should in this case be interpreted as a real-world functionality that is indistinguishable from an ideal one (in our case, a trusted third party). VIFF is a general software framework for developing secure multiparty computation. It provides software modules which abstract complex cryptographic and mathematical details into an easy way to use API for developers.

In our implementation, the VIFF framework is used to connect the talliers, and for executing the SMPC summation. First, the three talliers are launched. They generate key pairs, store the public key in a database and connect to each other. In the voting stage, an arbitrary number of ballots (but less than e) may be casted. The public keys are fetched from the database, and the ballots are constructed locally at the Voter Computer before these are inserted into the database. During the voting stage the talliers remain idle, waiting for a Remote Procedure Call (RPC) from the Authentication Server (AS) when the voting is complete. This invokes the counting procedure.

The scheme was implemented in a very small scale, with security parameter k as small as 20. Further work will enable for larger values for these variables, allowing more appropriate key sizes. The scheme can be extended to an arbitrary number of MPC talliers, instead of a fixed number as in this implementation, where three talliers were used.

8. CONCLUSIONS

We have shown how a ballot is sent as encrypted shares to multiple talliers. The talliers cannot individually reveal the content of the encrypted ballots. They must cooperate to jointly count the ballots. The counting process provides fairness since no single tallier can count or publish the tally.

We have presented a homomorphic multiparty counting with verification to work with the ballots according to the scheme of [11]. We present a variation of the Paillier cryptosystem for ballot construction for the verifiability of our counting process. Also, we show the secure multiparty computation for ballot counting. Finally, we implement the multiparty counting process on the VIFF platform.

The security proof of the modified Paillier cryptosystem and the verifiability of the counting process in the Paillier cryptosystem and the analysis on the practical computational efficiency are open problems.

ACKNOWLEDGEMENT

Supported by a grant from Iceland, Liechtenstein and Norway through the EEA Financial Mechanism and the Norwegian Financial Mechanism. This project is also co-financed from the state budget of the Slovak Republic.

REFERENCES

[1] SCHOENMAKERS, B.: *A Simple Publicly Verifiable Secret Sharing Scheme and its Application to Electronic Voting*, In Advances in Cryptology-CRYPTO 99, Vol. 1966 of Lecture Notes in Computer Science, Springer-Verlag, pp. 148-164, (1999).

[2] FUJIOKA, A. – OKAMOTO, T. – OHTA, K.: *A Practical Secret Voting Scheme for Large Scale Elections*, Advances in Cryptology - AUSCRYPT 92, LNCS volume 718, pp. 244-251. Springer Verlag, (1993).

[3] CRAMER, R. – FRANKLIN, M. – SCHOENMAKERS, B.: *Multi-authority Secret-Ballot Elections with Linear Work*, Advances in Cryptology - EUROCRYPT 96, LNCS volume 1070, pp. 72-83. Springer Verlag, (1996).

[4] CRAMER, R. – GENNARO, R. – SCHOENMAKERS, B.: *A Secure and Optimally Efficient Multi-authority Election Scheme*, in Proceedings of EUROCRYPT 97, LNCS series, volume 1233, pp. 103-118, (1997).

[5] BEN-OR, M. – GOLDWASSER, S. – WIGDERSON, A.: *Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation*, Proceedings of the eighth annual ACM Symposium on Principles of distributed computing, pp. 201-209, ISBN:0-89791-326-4, (1989).

[6] JURIK, M. J.: *Extensions to the Paillier Cryptosystem with Applications to Cryptological Protocols*, Ph.D. dissertation, BRICS Dissertation Series DS-03-9, ISSN 1396-7002, August, (2003).

[7] Ministry of Local Government and Regional Development: E-valg 2011-prosjektet (August 2008), Web: *http://www.regjeringen.no/nb/dep/krd/kampanjer/valg/-elektronisk stemmegivning.html?id=437385*, Accessed: November (2010).

[8] Ministry of Local Government and Regional Development: Forsøk med internettvalg i 2011 og bruk av pen kildekode (July 2009), Web: *http://www.regjeringen-.no/nb/dep/krd/kampanjer/valg/elektroniskstemmegivning/nytt-om-e-valg-2/nytt- om-e-valg/2009/forsok.html?id=570946*, Accessed: November (2010).

[9] ADIDA, B.: Helios. Web: *http://heliosvoting.org/*, Accessed: November (2010).

[10] GEISLER, M.: The Virtual Ideal Functionality Framework (VIFF). Web: *http://viff.dk/*

[11] BASED, M. A. – MJØLSNES, S. F.: "A Non-interactive Zero Knowledge Proof Protocol in an Internet Voting Scheme," in *Proceedings of the the 2nd Norwegian Security Conference (NISK 2009)*, pp. 148-160, ISBN: 978-82-519-2492-4, Tapir Akademisk Forlag, 2009.

[12] BASED, M. A. – REISTAD, T. I. – MJØLSNES, S. F.: "Internet Voting using Multiparty Computations," in *Proceedings of the the 2nd Norwegian Security Conference (NISK 2009)*, pp. 136-147, ISBN: 978-82-519-2492-4, Tapir Akademisk Forlag, 2009.

[13] BASED, M. A. – MJØLSNES, S. F.: "Universally Composable NIZK Protocol in an Internet Voting Scheme," in *Proceedings of the 6th International Workshop on Security and Trust Management*, Athens, Greece, September 23-24, 2010.

[14] PAILLIER, P.: *Public-Key Cryptosystems Based on Composite Degree Residuosity Classes*, Advances in Cryptology - EUROCRYPT 99, LNCS volume 1592, pp. 223-238, Springer Verlag, (1999).

[15] SHAMIR, A.: *How to Share a Secret*, Communications of the ACM, volume 22, issue 11, pp. 612-613. ACM Press, November, (1979).

[16] BLAKLEY, G. R.: *Safeguarding cryptographic keys*, In Proceedings of National Computer Conference, 48:313317, (1979).

Received November 5, 2010, accepted December 6, 2010

BIOGRAPHIES

Md. Abdul Based was born on 01. 01. 1979. In 2008 he graduated (MSc) in Information and Communication Systems Security at the Department of Computer and System Science at Royal Institute of Technology (KTH), Stockholm, Sweden. His Masters thesis title was "Information Handling in Security Solution Decisions". This thesis was part of the VRIEND (Value-Based Security Risk Mitigation in Enterprise Networks that are Decentralized) project. He is now in his 3rd year of PhD study at Norwegian University of Science and Technology (NTNU), Trondheim, Norway. His research topic is "Security Aspects of Internet Voting". His scientific research is focusing on information security.

Stig Fr. Mjølsnes received the Sivilingeniør degree in Physical Electronics in 1980, and the Dr. Ing. degree in 1990 both at the Norwegian Institute of Technology, Trondheim. The doctoral thesis was on the construction of cryptographic protocols for digital cash. From 2002, he holds a full professorship in information security at Department of Telematics at NTNU. His main professional interest is in the development of cryptographic protocols and security models that can be applied in society at large.

Marius Brekke Stenbek was born on January 18, 1985. Since 2006 he has studied for a 5 year MSc in Communication Technology at the Faculty of Information Technology, Mathematics and Electrical Engineering at the Norwegian University of Science and Technology (NTNU), where he will graduate in spring 2011. This paper was written in relation with his specialization project within Information Security at NTNU during the autumn of 2010.