Decryption phase in Norwegian electronic voting

Anders Smedstuen Lund and Martin Strand

Department of Mathematical Sciences, NTNU
ansmlu@gmail.com, martstr@math.ntnu.no

Abstract

We describe an efficient and secure decryption protocol to the Norwegian Internet voting project. We first adapt Groth's shuffle-decryption from 2010 to our purpose, and we prove all security properties in the random oracle model. We then describe the complete decryption algorithm, and prove that it maintains the security of the rest of the protocol.

Keywords: electronic voting protocols, verifiable shuffle, verifiable decryption.

1 Introduction

The Norwegian government ran remote Internet advance voting trials during the 2011 local election and the 2013 parliamentary election. In 2013, 77.3 % of the advance votes in the trial municipalities were electronic, and 36.4 % of the total number [18]. In June 2014, the minister for local government and modernisation cancelled any further trials [15]. However, in 2016 the system was resurrected for use in a number of local referendums, and is now being maintained by a private company.

The larger part of the protocol has been described and proven secure by Gjøsteen [7] and Gjøsteen and Lund [8]. In this paper, we introduce a new decryption protocol, and prove it to be secure in the random oracle model.

To fight voter coercion, the Norwegian Internet voting protocol allows each voter to submit as many ballots as he or she wishes, each cancelling any previous ballots. However, this implies that the ballot box service needs to store the voter's identity with the encrypted ballot. At the end of the voting period, the ballot box

Presented at the Norwegian Information Security Conference 2016 (NISK-2016).

selects the ballots that should be counted, removes the identities, and submits the encrypted ballots for decryption and counting.

In order not to reveal any information about the votes to the ballot box service, the decryption service should shuffle the ballots before publishing the decryption. This would usually be done through a mixnet. A proper mixnet requires several mix servers, and they should ideally be run by different organisations. Although the idea of a mixnet only requires a single trusted party, public trust in the system suggests that one should at least select organisations that one can trust with a certain high probability. Each organisation should also have sufficient competence to guarantee physical security requirements and proper uptime and maintenance. Furthermore, the ballots should not have to cross the country border at any point during the mixing. As cryptologists, we could trust our protocols, but we also need to make sure that the general population feel safe about the election. In a small country like Norway, this leaves a rather short list of candidate organisations. Therefore, we want a solution that does not use a full mixnet.

Finally, the decryption should be verifiable and zero knowledge with respect to the private key and the shuffle.

Related work

There exist numerous verifiable shuffles, that is, constructions to prove that a player has used a permutation π and some randomizers such that the output $\{c_i'\}$ decrypts to the same as some input ciphertexts $\{c_i\}$. Some are based on the fact that polynomials are stable under permutation of roots. The idea was first introduced by Neff in 2001 [13, 14], and has in particular been improved by Groth [9, 10].

The general idea is to prove that the permutation exists using arbitrary data, and then connect the arbitrary data to the problem in question. For ElGamal, one can use a variant of Chaum-Pedersen proofs to demonstrate knowledge of the randomisers.

This work is based on Groth's 2010 ideas [10]. Bayer and Groth [1] have done further work with this idea, and traded the size of the proof for more rounds and some more work for the prover.

Another line of research started by Furukawa and Sako in 2001 use permutation matrices [5, 4, 17, 20] instead of polynomials.

Most shuffles are tailored for reencrypting mixnets or decrypting mixnets. During the last years, the shuffles have become increasingly efficient, and are now practical.

Our contribution

We present a modification of Groth's shuffle-decryption of homomorphic encryptions [10]. In particular, we set the length of the mixnet to 1, and remove unnecessary computations for this case. Groth originally used a statistically hiding commitment scheme with the protocol. We have exchanged it for one that is computationally hiding and perfectly binding. We then prove the resulting one-move protocol to be computationally zero knowledge and unconditionally sound in the random oracle model and apply the Fiat-Shamir heuristic.

Next, we include our protocol in the Norwegian Internet voting protocol, and prove that the protocol is still secure.

Outline of the paper

Section 2 introduces the existing protocol, and gives a very brief outline of the instantiation. We also describe a variant of DDH which will be used in the security proof. Section 3 describes the verifiable shuffled decryption, the non-interactive version and proves all the main properties. In Section 4, we show that the protocol from the previous chapter fulfills the security requirements given by Gjøsteen [7]. Section 5 gives an estimate of the computational cost.

Acknowledgements

The authors wish to thank Kristian Gjøsteen for good discussions and sound advice. We also wish to thank our peer reviewers for a thorough job and helpful suggestions.

2 Background

This paper completes an open problem from Gjøsteen and Lund [7, 8], namely an analysed shuffle and decryption subprotocol. The original protocol, and especially the composition with the rest of the system was only supported by heuristic arguments. For simplicity, we use the same notation and security notions. We give a brief summary, but refer to the above cited papers for more details.

Let q and p be primes such that $p = 2q+1$ and let $G = (g) \subseteq \mathbb{Z}_p^*$ be the cyclic subgroup of quadratic residues modulo p. The group G has order q.

Consider the following subset of the instantiation of the cryptosystem:

- Let L be the number of options the voter can make on his ballot. The *key generation algorithm* $\mathcal{K}$ then chooses $2L$ random ElGamal private keys $a_{i,j}$ from $\mathbb{Z}_q^*$ with $i \in \{1,2\}$ and $j \in 1,\dots L$. Set $a_{3,j} = a_{1,j} + a_{2,j} \pmod q$, and compute corresponding public keys $y_{i,j} = g^{a_{i,j}}$ for $i \in \{1,2,3\}$. The key is split so that one can perform partial decryptions for the receipts. For brevity, let $dk_1 = (a_{11}, a_{12}, \dots, a_{1L})$, $dk_2 = (a_{21}, a_{22}, \dots, a_{2L})$ and $dk_3 = (a_{31}, a_{32}, \dots, a_{3L})$.

This notation is necessary for encoding the number of options available in Norwegian elections. They are, however, not important for the technical contribution of this paper. Therefore, we will simplify to just a and y whenever possible.

- The *encryption algorithm* $\mathcal{E}$ takes as input the encryption key ek, the voter identity V, and the ballot vector $\vec{v}$ from the set M^L of valid messages. It chooses $r \xleftarrow{r} \mathbb{Z}_q$ and computes $x = g^r$ and $w_i = y_{1i}^r v_i$, for $1 \leq i \leq L$, and then computes a proof $\pi_{\mathcal{E}}$ that the ciphertext is well-formed.

  It outpus the ciphertext $c = (V, x, \bar{x}, w_1, w_2, \dots, w_L, \pi_{\mathcal{E}})$, where $\bar{x}$ is some value generated similarly as x.

- The *deterministic extraction algorithm* $\mathcal{X}$ takes as input c. It verifies $\pi_{\mathcal{E}}$ and computes

$$\tilde{w} = \begin{cases} w_1 w_2 \cdots w_k & \text{if order is irrelevant,} \\ \prod_{i=1}^{k} w_i^i & \text{otherwise.} \end{cases}$$

  where k is the number of options used in the ballot. It outputs $\tilde{c} = (x, \tilde{w})$.

- The *decryption protocol* Π_{DP} which will be described below.

The following algorithm is only used in the security proof.

- Let $\phi(\vec{m}) = (m_1, m_2, \dots, m_L)$. The *anonymous decryption algorithm* $\mathcal{D}'$ takes as input $dk_1, \tilde{c}$ and outputs $\vec{m} = \phi(\tilde{w} x^{-dk_1})$.

The complete system consists of a number of voters, a ballot box, a receipt generator, the decryption service and an auditor. The auditor generally checks that all other parties adhere to the protocol, but in such a way that no secret information is leaked. In particular, this means that the auditor should check that the decryption and tally is correct, but without being able to correlate the ballots to the ciphertexts and identities he received from the ballot box and the receipt generator. We formalise this property later, and prove that it holds when we apply our shuffled decryption.

This paper is fairly technical, and assumes some background knowledge about complex protocols, zero knowledge proofs and commitment schemes. We give a brief informal introduction to some of the terms here, and refer to other texts for more details.

A zero knowledge proof must be *complete* (if both parties, the prover and the verifier, are honest, then the verifier should accept), *sound* (a cheating prover should cause the verifier to reject) and *zero knowledge* (a verifier should not be able to learn anything more than whether the relation in question holds or not). We typically restrict ourselves to *honest verifier* zero knowledge, in which the verifier must send truly random values, but might still be curious.

An important tool for such proofs are commitment schemes. These are used to bind a player to a message (the *binding* property), but without revealing the message to anyone else (the *hiding* property). Such schemes can have at most one of these properties unconditionally, while the other can only hold computationally.

3 Verifiable shuffled decryption

We will use Groth's shuffle of known contents (SKC) [10] as a basis for our protocol. The SKC protocol is unconditionally sound and computationally zero knowledge when used with a suitable commitment scheme. We will use a variation of Pedersen commitments [16]. Standard Pedersen commitments are perfectly hiding and computationally binding. To aid the composition of this protocol into the larger election protocol, we instead want a scheme that is perfectly binding.

Sample $g_1, \ldots, g_n$ and y at random from G. In order to commit to n messages $(m_1, \ldots, m_n)$, we select a random number r from $\mathbb{Z}_q$, and compute

$$\text{commit}(m_1, \ldots, m_n; r) = (g_1^{m_1+r}, \ldots, g_n^{m_n+r}, y^r)$$

Note that there can only be one opening for each commitment, so the commitment scheme is unconditionally binding and computationally hiding. In particular, this means that we can sample random exponents to compute the generators instead of sampling $g_1, \ldots, g_n$ and y directly, and hence we can use precomputation techniques on g to speed up the computation. We briefly return to this in Section 5. We sketch the security proof for this scheme in the appendix.

Groth's shuffle of known content is described in Figure 1a. The verifier provides a random point x, and the parties cooperate to construct a polynomial which proves that a commitment contains a permutation of the known data. One can later tie secrete data to the same commitment, to get a verifiable shuffle of any values. We note the following theorem.

Theorem 1 (Theorem 1, [10]). *The protocol in Figure 1a is a 4-move public coin special honest verifier zero-knowledge argument with witness-extended emulation*[1] *for c for being a commitment to a permutation of the messages* $m_1, \ldots, m_n$. *If the commitment scheme is statistically hiding, then the argument is statistical honest verifier zero-knowledge. If the commitment scheme is statistically binding, then we have unconditional soundness, i.e., the protocol is an SHVZK proof.*

The soundness is based on the difficulty of choosing a zero of a polynomial at random, so the forging probability in the case of a statistically binding is

[1]Witness-extended emulation is a stronger property than merely an knowledge extractor, and guarantees that one can output both a witness and a simulated transcript of the interaction, in expected polynomial time.

Public input: $c, m_1, \ldots, m_n$.
Private input to P: π and r such that
$c = \text{commit}(m_{\pi(1)}, \ldots, m_{\pi(n)}; r)$.

Prover | Verifier

$x \xleftarrow{r} \{0,1\}^\tau$

$\xleftarrow{\quad x \quad}$

$\alpha_1, \ldots, \alpha_n, r_\alpha, r_\Delta, r_p \xleftarrow{r} \mathbb{Z}_q$
$\Delta_1 \leftarrow \alpha_1, \Delta_2, \ldots, \Delta_{n-1} \leftarrow \mathbb{Z}_q, \Delta_n = 0$
$p_i \leftarrow \prod_{j=1}^{i} (m_{\pi(j)} - x)$
$c_\alpha \leftarrow \text{commit}(\alpha_1, \ldots, \alpha_n; r_\alpha)$
$c_\Delta \leftarrow \text{commit}(-\Delta_1\alpha_2, \ldots, -\Delta_{n-1}\alpha_n; r_\Delta)$
$c_p \leftarrow \text{commit}(\Delta_2 - (m_{\pi(2)} - x)\Delta_1 - p_1\alpha_2, \ldots,$
$\quad \Delta_n - (m_{\pi(n)} - x)\Delta_{n-1} - p_{n-1}\alpha_n; r_p)$

$\xrightarrow{\quad c_\alpha, c_\Delta, c_p \quad}$

$\beta \xleftarrow{r} \{0,1\}^\tau$

$\xleftarrow{\quad \beta \quad}$

$f_i \leftarrow \beta m_{\pi(i)} + \alpha_i$
$z \leftarrow \beta r + r_\alpha$
$f_{\Delta_i} \leftarrow \beta(\Delta_{i+1} - (m_{\pi(i+1)} - x)\Delta_i - p_i\alpha_{i+1})$
$\quad - \Delta_i\alpha_{i+1}$
$z_\Delta \leftarrow \beta r_p + r_\Delta$

$\xrightarrow{\quad f_1, \ldots, f_n, z, f_{\Delta_1}, \ldots, f_{\Delta_{n-1}}, z_\Delta \quad}$

Check group membership
$c^\beta c_\alpha \overset{?}{=} \text{commit}(f_1, \ldots, f_n; z)$
$c_p^\beta c_\Delta \overset{?}{=} \text{commit}(f_{\Delta_1}, \ldots, f_{\Delta_{n-1}}; z_\Delta)$
$F_1 \leftarrow f_1 - \beta x$
$\beta F_i \leftarrow F_{i-1}(f_i - \beta x) + f_{\Delta_{i-1}}$
$F_n \overset{?}{=} \beta \prod_{i=1}^{n} (m_i - x)$

(a) Shuffle of known content [10].

Public input: $\{(x_i, w_i) = (g^{r_i}, y^{r_i} m_i)\}$ and $\{z_i\}$.
Private input to P: π such that $z_i = m_{\pi(i)}$ and a such that $y = g^a$.

Prover | Verifier

$r, r_d, d_1, \ldots, d_n \xleftarrow{r} \mathbb{Z}_q$
$c \leftarrow \text{commit}(\pi(1), \ldots, \pi(n); r)$
$c_d \leftarrow \text{commit}(-d_1, \ldots, -d_n; r_d)$
$d_a, r_Z, d_Z, r_1, r_2 \xleftarrow{r} \mathbb{Z}_q$
$D \leftarrow g^{d_a}$
$Z \leftarrow g^{r_Z} \prod_{i=1}^{n} z_i^{-d_i}$
$C_1 \leftarrow \text{commit}(r_Z; r_1)$
$C_2 \leftarrow \text{commit}(d_Z; r_2)$

$\xrightarrow{\quad c, c_d, D, Z, C_1, C_2 \quad}$

$t_i \xleftarrow{r} \{0,1\}^\tau$

$\xleftarrow{\quad \{t_i\} \quad}$

$f_i \leftarrow t_{\pi(i)} + d_i$
$X \leftarrow g^{d_Z} \left(\prod_{i=1}^{n} x_i^{-t_i} \right)^{d_a}$

$\xrightarrow{\quad \{f_i\}, X \quad}$

$\lambda, e \xleftarrow{r} \{0,1\}^\tau$

$\xleftarrow{\quad \lambda, e \quad}$

$\xleftrightarrow{\text{SKC: } c^\lambda c_d \text{commit}(f_1, \ldots, f_n; 0)}$
$= \text{commit}(\lambda\pi(i) + t_{\pi(i)}; \rho)$

$f \leftarrow ea + d_a$
$f_Z \leftarrow er_Z + d_Z$
$z_Z \leftarrow er_1 + r_2$

$\xrightarrow{\quad f, f_Z, z_Z \quad}$

Check group membership
$Z^e X \prod_{i=1}^{n} x_i^{f t_i} w_i^{-t_i e} z_i^{f_i e} \overset{?}{=} g^{f_Z}$
$y^e D \overset{?}{=} g^f$
$C_1^e C_2 \overset{?}{=} \text{commit}(f_Z; z_Z)$

(b) Verifiable shuffled decryption.

bounded by $(n-1) \cdot 2^{-\tau}$, where τ is some parameter set to achieve sufficiently strong soundness.

Here, we use the subprotocol as a necessary primitive, and refer to the original work for more details.

We now proceed to describe a protocol for shuffled decryption. It is similar to Groth's protocol for shuffle-decryption, i.e. for mix-nets with several servers for mixing and partly decryption. Note that the first challenges in the SKC can be sent with the second round of challenges in the protocol in Figure 1b.

The basic idea is similar to Groth's shuffle-decryption over many nodes, in that we tie our data to the commitments of known data, and prove that we know the values that have been used to transform the data. However, this protocol is slightly simpler, since the second part of the public input is a set of plaintexts, and not new ciphertexts. Therefore, we don't have to prove relations with the first coordinate of the ElGamal tuples. Instead, the prover must demonstrate that he knows the decryption key, and that is has been used to decrypt under the permutation otherwise proven to exist. The key element is then the response f, which proves knowledge of the secret key a, which then again is applied to the first coordinate of the original ciphertexts, causing them to cancel the obfuscation of the second coordinate.

We prove this secure in the Random Oracle Model. The proofs of the following propositions can be found in the appendices.

Proposition 1 (Soundness)**.** *The protocol for verifiable shuffled decryption is unconditional sound.*

We also achieve good privacy.

Proposition 2 (Zero knowledge)**.** *The protocol for verifiable shuffled decryption is SHVZK.*

We finally make this protocol non-interactive by applying the Fiat-Shamir transformation [3].

4 Completing the e-voting cryptosystem

We now describe the decryption protocol Π_{DP} and prove that it satisfies the the security notions in the Norwegian e-voting project. This protocol completes the instantiation described by Gjøsteen and Lund in [8].

We define Π_{DP} as follows: The auditor A and the decrypter D both receive or compute $\tilde{c}_i = \mathcal{X}(V_i, c_i)$. In addition, D also receives the decryption key dk_1. The decrypter chooses an random permutation π and computes $z_i = \mathcal{D}'(dk_1, \tilde{c}_{\pi(i)})$ for all i. He then sends the decryption to the auditor, and the parties run the protocol for verifiable shuffled decryption.

There are one completeness notion and two security notions from the Norwegian e-voting project not proved in Lund and Gjøsteen's description because the decryption protocol is missing. We present these notions and prove them for our decryption protocol.

C2. *For any sequence of messages, encrypting, extracting and then running the decryption protocol should faithfully reproduce the messages, up to the action of the order map.*

For any message and identity sequences $\vec{m}_1, \vec{m}_2, \ldots, \vec{m}_n$ and $V_1, V_2, \ldots, V_n$, if the following actions happen:

> $(ek, dk_1, dk_2, dk_3) \leftarrow \mathcal{K}$; for i from 1 to n: $c_i \leftarrow \mathcal{E}(ek, V_i, \vec{m}_i)$, $\tilde{c}_i \leftarrow \mathcal{X}(V_i, c_i)$; the protocol Π_{DP} is run with ek and $(\tilde{c}_1, \ldots, \tilde{c}_n)$ as public input and dk_1 as the prover's private input.

Then the prover and verifier in the protocol both output the same sequence of messages, and that sequence is a permutation of the sequence $\omega(\vec{m}_1), \ldots, \omega(\vec{m}_n)$.

Since ElGamal is homomorphic, the zero-knowledge proofs are complete and ϕ recovers a proper ballot from the product, the completeness requirement will hold.

A-**Privacy** *An adversary that runs the verifier part of the decryption protocol should not be able to correlate ciphertexts with decryptions.* We play the following game between a simulator and an adversary, and the probability that the adversary wins should be close to 1/2.

A simulator samples $b \leftarrow \{0,1\}$ and computes $(ek, dk_1, dk_2, dk_3) \leftarrow \mathcal{K}$. The adversary gets ek, then chooses two sequences of identities $V_1, \ldots, V_{n'}$, $V'_1, \ldots, V'_{n''}$ and corresponding messages $\vec{m}_1, \ldots, \vec{m}_{n'}$, $\vec{m}_1^{(0)}, \ldots, \vec{m}_{n''}^{(0)}$, for some $n', n'' < n$.

The simulator sets π_0 to be the identity map on $\{1, 2, \ldots, n'\}$, and samples a random permutation π_1 on $\{1, 2, \ldots, n'\}$ and a sequence of random messages $\vec{m}_1^{(1)}, \ldots, \vec{m}_{n''}^{(1)}$. Then the simulator computes $c'_i \leftarrow \mathcal{E}(ek, V'_i, \vec{m}_i^{(b)})$ for $i = 1, 2, \ldots, n''$ and $c_i \leftarrow \mathcal{E}(ek, V_i, \vec{m}_{\pi_b(i)})$, $\tilde{c}_i \leftarrow \mathcal{X}(V_i, c_i)$ for $i = 1, 2, \ldots, n'$, sends $c'_1, \ldots, c'_{n''}, c_1, \ldots, c_{n'}$ to the adversary and runs the prover part of the protocol Π_{DP} with appropriate input against the adversary's verifier.

Finally, the adversary outputs $b' \in \{0,1\}$ and wins if $b = b'$.

The first sequence of messages models the votes that will be counted, with a connection to voter identities. If $b = 1$, this connection will be broken. The other sequence of messages are those that will be discarded before decryption, and the $b = 1$ case replaces those completely. If the adversary has no advantage in this game, he cannot say anything about who voted what, and the content of the discarded votes.

Proposition 3 (A-Privacy)**.** *Assume that an adversary bounded by χT against Decision Diffie-Hellman has advantage at most ϵ_{DDH} and $\chi T < 2^{\tau/2} - 1$, then any adversary against A-privacy using time at most T has an advantage of at most $(2Tn + 4T + 2n)2^{-\tau} + 3\epsilon_{DDH}$, where n is the number of ballots to be counted.*

Game 1 We begin with the A-privacy game between a simulator and an adversary. The game requires time at most T and the adversary submits at most n ballots.

If E_1 is the event that the adversary correctly guesses the bit b, the adversary's advantage is

$$\epsilon = |\Pr[E_1] - 1/2|.$$

Game 2 We now simulate the verifiable shuffled decryption. This can fail if either the reprogramming of the oracle fails (with probability $4T \cdot 2^{-\tau}$), or if the adversary can break the hiding property of the commitment scheme with probability ϵ_{DDH}. The event of losing the game is F_2, and we have

$$|\Pr[F_2]| \leq 4T \cdot 2^{-\tau} + \epsilon_{DDH}.$$

Game 3 We modify the game such that the simulator now selects π at random, and outputs $z_i = m_{\pi(i)}$ instead of $z_i = \mathcal{D}'(dk_1, \tilde{c}_{\pi(i)})$. If $\Pr[E_3]$ is the event that adversary correctly guesses the bit b, then $|\Pr[E_3] - \Pr[E_1]| \leq 4T \cdot 2^{-\tau} + \epsilon_{DDH}$

Game 4 Next we use the simulator Sim^I_{eqdl} described by Gjøsteen and Lund [8, Section 3.1] to simulate the validity proofs of the ballots, with an according oracle reprogramming. Then

$$|\Pr[E_3] - \Pr[E_4]| \leq \frac{2Tn}{2^\tau}$$

Game 5 We now randomise $c_1, \dots, c_{n'}, c'_1, \dots, c'_{n''}$ by using random group elements instead of real encryptions. By Gjøsteen and Lund [8, Lemma 4.2], we get

$$|\Pr[E_5] - \Pr[E_4]| \leq 2(n2^{-\tau} + \epsilon_{DDH}).$$

This game does not include any information on the bit b, so $\Pr[E_5] = 1/2$.

By the triangle inequality, we get

$$\epsilon \leq (2Tn + 4T + 2n)2^{-\tau} + 3\epsilon_{DDH}$$

***D*-Integrity** *An adversary that runs the prover's part of the protocol* Π_{DP} *should not be able to tamper with the decryption.* We play the following game between a simulator and an adversary, and the probability that the adversary wins should be close to 0.

A simulator computes $(ek, dk_1, dk_2, dk_3) \leftarrow \mathcal{K}$. The adversary gets ek and dk_1, then chooses a sequence of identities $V_1, \dots, V_n$ and messages $\vec{m}_1, \dots, \vec{m}_n$.

The simulator computes $c_i \leftarrow \mathcal{E}(ek, V_i, \vec{m}_i)$, $\tilde{c}_i \leftarrow \mathcal{X}(V_i, c_i)$, $i = 1, 2, \dots, n$, sends $\tilde{c}_1, \dots, \tilde{c}_n$ to the adversary and runs the verifier part of the protocol Π_{DP} with appropriate input against the adversary's prover.

The adversary wins if the verifier run outputs a sequence of messages that is not a permutation of $\omega(\vec{m}_1), \dots, \omega(\vec{m}_n)$.

Proposition 4 (D-Integrity)**.** *Assume that the number of random oracle queries from the adversary is limited by time T. Then the probability of the adversary winning is close to* 0*.*

This follows directly since the protocol for verifiable shuffled decryption is unconditionally sound.

5 Runtime

Our proposed protocol for verifiable shuffled decryption is efficient, and in particular when we apply techniques for precomputation and multiexponentiation. This section is only meant to give a conservative estimate of the real runtime, given by the number of full square-and-multiply exponentiations, so that one can compare it against other methods. We keep the analysis simple to ease such comparison; a more thorough optimisation would most probably result in an even better result, but so would also be the case for alternative solutions.

According to [12, 11], we can save about 80 % of online computing time with precomputation and 50 % with multiexponentiation when applicable.

The prover needs to compute 5 full-length commitments, with a total of $5n + 5$ exponentiations, to an estimated cost of about n full exponentiations. In order to compute Z, the prover must do a multiexponentiation of length n, which costs approximately $0.5n$. Finally, there are n short exponentiations, at 10 % of the cost. In total, the prover's cost is $1.6n$.

The verifier have to use ordinary techniques to compute his 2 commitments. Also, there are $2n$ short exponentiations and n full exponentiations that can be computed with multiexponentiation, giving the equivalent of $2.7n$ full exponentiations.

The cost is slightly better than just performing a verifiable shuffle followed by an ordinary protocol for verifiable decryption. One can achieve a lower cost by also accepting weaker security properties. For a full comparison, see Strand [19]

References

[1] Stephanie Bayer and Jens Groth. Efficient zero-knowledge argument for correctness of a shuffle. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology - EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 263–280. Springer, 2012.

[2] Ivan Damgård, Kasper Dupont, and Michael Østergaard Pedersen. Unclonable group identification. In Serge Vaudenay, editor, *EUROCRYPT*, volume 4004 of *Lecture Notes in Computer Science*, pages 555–572. Springer, 2006.

[3] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, 1986.

[4] Jun Furukawa. Efficient and verifiable shuffling and shuffle-decryption. *IEICE Transactions*, 88-A(1):172–188, 2005.

[5] Jun Furukawa and Kazue Sako. An efficient scheme for proving a shuffle. In Joe Kilian, editor, *CRYPTO*, volume 2139 of *Lecture Notes in Computer Science*, pages 368–387. Springer, 2001.

[6] Kristian Gjøsteen. A latency-free election scheme. In Tal Malkin, editor, *CT-RSA*, volume 4964 of *Lecture Notes in Computer Science*, pages 425–436. Springer, 2008.

[7] Kristian Gjøsteen. The Norwegian internet voting protocol. Cryptology ePrint Archive, Report 2013/473, 2013. `http://eprint.iacr.org/`.

[8] Kristian Gjøsteen and Anders Smedstuen Lund. The Norwegian internet voting protocol: A new instantiation. Cryptology ePrint Archive, Report 2015/503, 2015. `http://eprint.iacr.org/`.

[9] Jens Groth. A verifiable secret shuffle of homomorphic encryptions. In Yvo Desmedt, editor, *Public Key Cryptography*, volume 2567 of *Lecture Notes in Computer Science*, pages 145–160. Springer, 2003.

[10] Jens Groth. A verifiable secret shuffle of homomorphic encryptions. *J. Cryptology*, 23(4):546–579, 2010.

[11] Chae Hoon Lim. Efficient multi-exponentiation and applications to batch verification of digital signatures. Available at `http://dasan.sejong.ac.kr/~chlim/pub/multi_exp.ps.`, 2000.

[12] Chae Hoon Lim and Pil Joong Lee. More flexible exponentiation with precomputation. In Yvo Desmedt, editor, *CRYPTO*, volume 839 of *Lecture Notes in Computer Science*, pages 95–107. Springer, 1994.

[13] C. Andrew Neff. A verifiable secret shuffle and its application to e-voting. In *Proceedings of the 8th ACM conference on Computer and Communications Security*, CCS '01, pages 116–125, New York, NY, USA, 2001. ACM.

[14] C. Andrew Neff. Verifiable mixing (shuffling) of ElGamal pairs. Technical report, In proceedings of PET '03, LNCS series, 2003.

[15] Ministry of Local Government and Modernisation. Internet voting pilot to be discontinued. Internet, jun 2014. `https://www.regjeringen.no/en/aktuelt/Internet-voting-pilot-to-be-discontinued/id764300/`.

[16] Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Proceedings of the 11th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '91, pages 129–140, London, UK, UK, 1992. Springer-Verlag.

[17] Kun Peng, Ed Dawson, and Feng Bao. Modification and optimisation of a shuffling scheme: stronger security, formal analysis and higher efficiency. *Int. J. Inf. Sec.*, 10(1):33–47, 2011.

[18] Signe Bock Segaard, Dag Arne Christensen, Bjarte Folkestad, and Jo Saglie. *Internettvalg*. Institutt for samfunnsforskning, 2014. English summary available at `https://www.regjeringen.no/globalassets/upload/kmd/komm/rapporter/isf_internettvalg_english-summary.pdf`.

[19] Martin Strand. Verifiable shuffled decryption, 2013. Master thesis. Available at `https://brage.bibsys.no/xmlui/handle/11250/259169`.

[20] Björn Terelius and Douglas Wikström. Proofs of restricted shuffles. In Daniel J. Bernstein and Tanja Lange, editors, *AFRICACRYPT*, volume 6055 of *Lecture Notes in Computer Science*, pages 100–113. Springer, 2010.

Appendix A Security of the commitment scheme

In order to prove that the commitment scheme in Section 3 is computationally hiding, consider the following problem which is equivalent to the DDH problem [7, 2, 6].

L-DDH. *Given $(g_0, \dots, g_L) \in G^{L+1}$ (where at least $g_1, \dots, g_L$ are sampled at random), decide if $(x_0, \dots, x_L) \in G^{L+1}$ was sampled uniformly from the set $\{(g_0^s, \dots, g_L^s) \mid 0 \leq s < q\}$ or uniformly from G^{L+1}.*

We now present a distinguisher based on the commitment system. Assume that $(g_0, \dots, g_L)$ are given, and that we want to decide whether $(x_0, \dots, x_L)$ is a real Diffie-Hellman tuple. Send $(g_0, \dots, g_{L-1}; g_L)$ to the adversary, and wait for a set $m_0, \dots, m_{L-1}$ of messages. Now compute

$$c \leftarrow (g_0^{m_0} x_0, \dots, g_{L-1}^{m_{L-1}} x_{L-1}; x_L)$$

and send c to the adversary.

Note that there exists an r such that $x_L = g_L^r$, so if $(x_0, \dots, x_L)$ is really a Diffie-Hellman tuple, then c is a commitment to $m_0, \dots, m_{L-1}$. If not, then c is a commitment to random messages. Our distinguisher have the same advantage ϵ_{DDH} as the adversary, on the expense of L extra products.

Appendix B Proof of Proposition 1

Proof. We want to prove that there exists a permutation π such that $m_{\pi(i)} = z_i$. We first note that the shuffle of known content is unconditionally sound, so the prover must have π and ρ such that $c^\lambda c_d \operatorname{commit}(f_1, \dots, f_n; 0) = \operatorname{commit}(\lambda\pi(i) + t_{\pi(i)}; \rho)$. Since λ is random, we have, except with probability $2^{-\tau}$, that $c = \operatorname{commit}(\pi(1), \dots, \pi(n); r)$ for some r. Assume that there exist $-d_1, \dots, -d_n$, then

$$\operatorname{commit}(-d_1, \dots, -d_n; r_d) \operatorname{commit}(f_1, \dots, f_n; 0) = \operatorname{commit}(t_{\pi(1)}, \dots, t_{\pi(n)}; r_d)$$

for some r_d. Therefore, we must have $f_i = t_{\pi(i)} + d_i$.

All the final checks held, so f, f_Z and z_Z must have been well-formed. This allows us to use them in the following computations. Note that one can extract a from f by rewinding.

Note the equation

$$Z^e X \prod_{i=1}^{n} x_i^{f t_i} w_i^{-t_i e} z_i^{f_i e} \overset{?}{=} g^{f_Z} \tag{1}$$

which we can rewrite into two parts: One that depends on the exponent e, and one that doesn't.

$$\begin{aligned}
1 &= g^{-fz} Z^e X \prod_{i=1}^{n} x_i^{ft_i} w_i^{-t_i e} z_i^{f_i e} \\
&= g^{-erz-dz} Z^e X \prod_{i=1}^{n} x_i^{(ae+d_a)t_i} w_i^{-t_i e} z_i^{(t_{\pi(i)}+d_i)e} \\
&= g^{-erz-dz} Z^e X \prod_{i=1}^{n} x_i^{(ae+d_a)t_i} (x_i^a m_i)^{-t_i e} z_i^{(t_{\pi(i)}+d_i)e} \\
&= \left(Z g^{-rz} \prod_{i=1}^{n} \left(m_{\pi(i)}^{-1} z_i \right)^{t_{\pi(i)}} z_i^{d_i} \right)^e X g^{-dz} \left(\prod_{i=1}^{\tau} x_i^{t_i} \right)^{d_a}
\end{aligned}$$

All other values above are fixed when e is chosen, so both parts must be 1 with overwhelming probability. The part that doesn't depend on e shows that X is well-formed.

Since e is arbitrary, we must with probability $1-2^{-\tau}$ have

$$Z g^{-rz} \prod_{i=1}^{n} \left(m_{\pi(i)}^{-1} z_i \right)^{t_{\pi(i)}} z_i^{d_i} = 1$$

This again splits into two parts, where one is dependent of $\{t_i\}$, while the other is fixed after the first round. If not both parts equal 1, a dishonest P^* must have guessed $\{t_i\}$ such that a particular group element was hit. This only happens with probability $\frac{1}{q}$.

$$Z g^{-rz} \prod_{i=1}^{n} z_i^{d_i} = 1$$

$$\prod_{i=1}^{n} \left(m_{\pi(i)}^{-1} z_i \right)^{t_{\pi(i)}} = 1$$

which implies that Z is well-formed, and that $z_i = m_{\pi(i)}$. The probability of P^* winning is then at most

$$\frac{n+1}{2^\tau} + \frac{2}{2^\tau} + \frac{1}{q} < \frac{n+4}{2^\tau}.$$

□

Appendix C Proof of Proposition 2

Proof. We prove the proposition by describing a simulator that is able to produce a convincing transcript without using dk_1. For completeness, we include the shuffle of known content in the argument.

1. Select x and β at random. Also select f_i for $i = 1, \ldots n$, z, F_i for $i = 2, \ldots n-1$ and z_Δ at random.
2. Set $F_1 \leftarrow f_1 - x\beta$ and $F_n \leftarrow x \prod_{i=1}^{n}(m_i - x)$. Compute f_Δ accordingly.
3. Compute $c_a \leftarrow \mathrm{commit}(0, \ldots, 0; r)$.
4. Compute c_α and c_Δ to fit.

We denote this sampling by

$$(c_\alpha, c_\Delta, c_p, f_1, \ldots, f_n, z, f_{\Delta_1}, \ldots, f_{\Delta_{n-1}}, z_\Delta) \leftarrow \mathit{Sim}_{skc}(aux, g, c, m_1, \ldots, m_n, x, \beta).$$

This is indistinguishable from a real transcript. First recall that the commitment scheme is computationally hiding. Therefore, a computationally bounded adversary cannot distinguish real and simulated commitments. Now, the real and simulated f_i, z and z_Δ are indistinguishable since the real elements contain a random element not used anywhere else, while the simulated values are random as well, and so the distributions are equal. Finally, note that f_Δ is indistinguishable from the real construction since the checks on the commitments and the construction of F_i holds.

We now describe a simulator for the shuffled decryption.

1. Select λ at random. Select a random permutation π, and compute c using the random permutation. Compute c_d according to the real protocol.
2. Select e at random. Also select C_1, Z, f, f_Z and z_Z at random.
3. Compute D and C_2 such that $y^e D = g^f$ and $C_1^e C_2 = \mathrm{commit}(f_Z; z_Z)$.
4. Query the oracle for t_i for $i = 1, \ldots n$.
5. Compute X and f_i such that $X = g^{dz} \left(\prod_{i=1}^{n} x_i^{-t_i}\right)^{d_a}$ and $f_i = t_{\pi(i)} + d_i$.
6. Simulate the shuffle of known contents, Sim_{skc}.

Let Sim_{vsd} be

$$\begin{aligned}(c, c_d, D, Z, C_1, C_2, f_1, \ldots f_n, X, f, f_Z, z_Z)& \\ \leftarrow \mathit{Sim}_{vsd}(aux, g, (x_1, w_1), \ldots, (x_n, w_n), z_1, \ldots, z_n, \lambda, e).&\end{aligned}$$

As above, we use that the commitment scheme is computationally hiding, and that we can simulate the shuffle of known content. Therefore, all simulated commitments are indistinguishable from the real commitments. Next, f, f_Z and z_Z all contain randomness not repeated anywhere else, and so they are indistinguishable from random numbers. The simulation of D and X ensure that they are also indistinguishable. We produce X_i as $t_{\pi(i)} + d_i$. Only the permutation differs, and since the shuffle of known content is simulated, nobody can distinguish. Finally, we are left with $Z = g^{r_Z} \prod_{i=1}^{n} z_i^{-d_i}$. Note that r_Z is no longer used in any other commitments, and so Z is indistinguishable from a random group element. $\square$