Verifiable Mixing (Shuffling) of ElGamal Pairs

C. Andrew Neff*

April 21, 2004†

Abstract

We give an improved presentation of the protocol first published in [23]. That paper contained some minor misprints, and was thin on formal proofs of correctness; both shortcomings are addressed in the current version. We also make some brief comparisons with, and criticisms of some more recent publications which build on the results presented in [23]. In particular, we point out that, in contrast to other protocols, the protocol of this paper is unconditionally sound.

1 Introduction

In [23], one of the first efficient protocols for verifiably *shuffling*, or *mixing* a list of encrypted elements was given. In that paper, two different types of mixes were considered. In one, the elements to be permuted were individual group elements, while the second setting considered lists of ElGamal pairs. The protocol in both cases was similar, and in both presentations, a multiplicative factor (in the exponent domain) was inadvertently omitted, giving the impression that the protocols might not be correct.

In this paper, we correct the missing factor as well as eliminate some redundant operations. We also include details of the formal correctness proofs that were only sketched in the original. For the sake of brevity, we consider

*aneff@votehere.net

†This work is still in progress.

only the ElGamal case. This is the setting most commonly found in electronic voting applications. The modifications required for mixing of group elements are quite straightforward, and will be documented in a future work.

Recently, there have been other papers on the subject, and it is worth making some brief comparisons. The original work ([23]) was founded on three novel ideas:

1. Construct a general mix protocol from one that could only be constructed for input with "known exponents" (known by the prover, that is).
2. Construct a "known exponent mix" from two principles:
   (a) Evaluation of a pair of polynomials at a common random point.
   (b) A generalization of the Chaum-Pedersen protocol to a protocol for proving statements about the product of secret exponents.

Groth ([19]) uses the same approach to construct a verifiable mix protocol. Similarities are somewhat obscured by notation, since Groth has chosen to present the ideas in an abstract homomorphic group setting. There are some shortcomings of the form of the protocol presented there that are worth noting however:

1. Groth explicitly relies on the assumption that $\log_g h$ is unknown to the prover. While this assumption may be reasonable in some cases, when applied to the multi-authority election setting, it results in an election protocol that is *not universally verifiable.* This is because a subset of authorities can collude to reconstruct the election private key, and then use this information to forge one or more of their shuffle transcripts. The criteria of universal verifiability requires that even if *all* authorities collude, they are still prevented – at least by computational constraints – from altering election results.

   It is possible for Groth to fix this problem, but at the expense of introducing more exponentiations (we expect $3k$). Conversely, the exponentiation count in this paper can be reduced if the unknown logarithm assumption is used. (Specifically, equations 22 and 23, 31 and 32, 34 and 35 can be combined in pairs.)
2. Groth also relies on a basis of group elements for which whose logarithms (to a common base) the prover can not find non-trivial linear relations. The use of such a basis is undesirable on two counts:

(a) A protocol based on this assumption can be at best computationally sound.

(b) Reliance on such a basis "hides complexity". In typical application, the construction of these elements involves exponentiations in order to project into the encoding subgroup. These exponentiations are typically "large," costing a few times that of a exponentiation in the protocol itself. One can make an argument that these exponentiations can be amortized over several mixes. However, so too can some of the exponentiations in the protocol of this paper. A fair comparison of complexity should carefully set aside, and count separately, "precomputation steps."

The protocol version we present is *unconditionally sound.* That is, even unlimited computational resources do not help the prover forge a transcript with an interactive verifier. While a non-interactive transcript *may* still be forgeable with unlimited computational power, even this possibility can be removed by introducing enough "parallel executions." It is only computational zeroknowledge, which, in general is less desirable than honest-verifier zeroknowledge. However, in this case, the difference is irrelevant: The original secret is only protected by the same computational limitation. Thus, it would be more effective for an adversary who wishes to break the secret to attack the data that is input to the protocol, rather than the protocol itself. In short, in this case, it seems undesirable to sacrifice unconditional soundness for the sake of achieving honest-verifier zeroknowledge.

2 Notation

In the following, unless explicitly stated otherwise, n will be a positive integer, p and q will be prime integers, publicly known. Arithmetic operations are performed in the modular ring $\mathbf{Z}_p$ (or occasionally $\mathbf{Z}_n$), and $g \in \mathbf{Z}_p$ will have (prime) multiplicative order q. So, trivially, $q \mid (p-1)$. We further assume that p has been chosen so that $(q^2, (p-1)) = q$. This means that the multiplicative group $\mathbf{Z}_p^*$ has a *unique* order q subgroup – a fact that we will henceforth generally take for granted. We denote this subgroup by $\mathcal{G}$. Finally, we use g to denote a fixed generator of $\mathcal{G}$, which has been publicly accepted (i.e. agreed upon).

In each proof protocol, we refer to the prover (shuffler, or mixer) as $\mathcal{P}$, and to the verifier (auditor) as $\mathcal{V}$.

We recall the Chaum-Pedersen proof of equality for discrete logarithms ([11]). For $G, X, H, Y \in \mathbf{Z}_p$ this is a proof of knowledge for the relation

$$\log_G X = \log_H Y \tag{1}$$

It is not known to be zero-knowledge, however it is known to be honest-verifier zeroknowledge. These are sufficient for our main application where the verifier is implemented via the Fiat-Shamir heuristic. (See [16] and [8].)

The main result of this paper uses a sub-protocol which can be viewed as a natural multi-variable generalization of the Chaum-Pedersen protocol. This construction was originally given in in [23], and referred to there as the *Iterated Logarithmic Multiplication Proof Protocol (ILMPP)*. For the sake of brevity, we will not discuss the construction itself, or how to see it as a multi-variable generalization of Chaum-Pedersen in this paper. The interested reader may find this discussion in [23].

Following standard notation, we let $\mathbf{Z}_n$ be the ring of integers mod n, $\mathbf{Z}_n[x]$ be the ring of polynomials over $\mathbf{Z}_n$, $\mathbf{Z}_n^k$ be the standard k-dimensional vector space over $\mathbf{Z}_n$, and Σ_k the group of permutations on $\{1, \ldots, k\}$. In addition, we refer to the following which is less standard:

Definition 1 For $\vec{a} = (a_1, \ldots, a_k) \in \mathbf{Z}_p^k$, and $\pi \in \Sigma_k$, define

$$\vec{a}_\pi \ = \ (a_{\pi(1)}, \ldots, a_{\pi(k)})$$

We note the following collection of well know results, since they will be heavily used in the remainder of the paper. (See [21] and [33].)

Lemma 1 *Let $f(x) \in \mathbf{Z}_q[x]$, be a polynomial of degree d. Then there are at most d values $z_1, \ldots, z_d \in \mathbf{Z}_q$ such that $f(z_i) = 0$.*

Corollary 1 *Let $f(x)$, $g(x) \in \mathbf{Z}_q[x]$ be two* monic *polynomials of degree at most d, with $f \neq g$. Then there are at most $d - 1$ values $z_1, \ldots, z_{d-1} \in \mathbf{Z}_q$ such that $f(z_i) = g(z_i)$.*

Corollary 2 *Let $f(x)$, $g(x) \in \mathbf{Z}_q[x]$ be two* monic *polynomials of degree at most d, with $f \neq g$. If $t \in_R \mathbf{Z}_q$ (t is selected at random from $\mathbf{Z}_q$), then*

$$P(\{t : f(t) = g(t)\}) \ \leq \ \frac{d-1}{q}$$

Lemma 2 *Suppose $v = (v_1, \ldots, v_k)$ and $w = (w_1, \ldots, w_k)$ are both elements of $\mathbf{Z}_q^k$, and that $H \subset \mathbf{Z}_q^k$ is a hyperplane through the origin (i.e. $(0, \ldots, 0) \in H$). If $v \notin H$, there is exactly one element, $c \in \mathbf{Z}_q$ satisfying*

$$w \; - \; c\,v \;\; \in \;\; H \tag{2}$$

Corollary 3 *If v, w and H are as in lemma 2, and if c is generated from $\mathbf{Z}_q$ at random, then*

$$P(\{w \; - \; c\,v \;\; \in \;\; H\}) \; = \; \frac{1}{q} \tag{3}$$

Lemma 3 *Fix $v = (v_1, \ldots, v_k) \in \mathbf{Z}_q^k$, $v \neq 0$, and $a \in \mathbf{Z}_q$. If $r \in_R \mathbf{Z}_q^k$ is chosen at random, then*

$$P(\{r : v \cdot r = a\}) \; = \; \frac{1}{q}$$

Corollary 4 *Fix $v = (v_1, \ldots, v_k) \in \mathbf{Z}_q^k$, $v \neq 0$. If $r \in_R \mathbf{Z}_q^k$ is chosen at random, then*

$$P(\{r : v \cdot r = 0\}) \; = \; \frac{1}{q}$$

Definition 2 Let

$$\begin{aligned} \Delta_q^k \; &\doteq \; \{\, (v_1, \ldots, v_k) \in \mathbf{Z}_q^k \; : \; v_i = v_j \text{ for some } 1 \leq i \neq j \leq k \,\} \\ \bar{\Delta}_q^k \; &\doteq \; \mathbf{Z}_q^k \; - \; \Delta_q^k \\ \bar{\Delta}_q^{k+} \; &\doteq \; \bar{\Delta}_q^k \cap \mathbf{Z}_q^{*k} \end{aligned}$$

Lemma 4 *Fix $v = (v_1, \ldots, \ldots v_k) \in \mathbf{Z}_q^k$, $v \neq 0$, and $a \in \mathbf{Z}_q$. If $r \in_R \bar{\Delta}_q^{k+}$ is chosen at random (uniform distribution on $\bar{\Delta}_q^{k+}$), and if $k < q/2$, then*

$$P(\{\, r : v \cdot r = a\,\}\,) \; < \; \frac{2}{q}$$

This estimate can be improved considerably, but it will suffice for our application.

Definition 3 Let (X, Y) be an ElGamal pair with encryption parameters (g, h), where $h = g^s$ with s "secret". We say that the *message*, m, of (X, Y) is Y/X^s.

3 The Simple k-Shuffle

The first shuffle proof protocol we construct requires a restrictive set of conditions – in short, that the prover knows the logarithms of the elements to be permuted. It's importance lies in the fact that it serves as an essential step in the more general protocol to follow.

Definition 4 Suppose that G and Γ, and two sequences $X_1, \ldots, X_k$, and $Y_1, \ldots, Y_k$ are all publicly known elements of $\mathcal{G} \subset \mathbf{Z}_p^*$. Suppose also, that the prover, $\mathcal{P}$, knows $\gamma = \log_G \Gamma$, $x_i = \log_G X_i$ and $y_i = \log_G Y_i$, for all $1 \leq i \leq k$, where G is some generator of $\mathcal{G}$, but that all these logarithms are unknown to $\mathcal{V}$.

$\mathcal{P}$ is required to convince $\mathcal{V}$ that there is some permutation, $\pi \in \Sigma_k$ with the property that, for all $1 \leq i \leq k$,

$$Y_i = X_{\pi(i)}^{\gamma} \tag{4}$$

without revealing any information about x_i, y_i, γ, or π.

We call this problem the *Simple k-Shuffle Problem.* It can be solved as follows:

3.1 Simple k-Shuffle Proof Protocol

SSA. 1. $\mathcal{V}$ generates a random $t \in \mathbf{Z}_q$, and gives t to $\mathcal{P}$ as a challenge.

SSA. 2. $\mathcal{P}$ secretly computes

$$\begin{aligned} \hat{x}_i &= x_i - t \quad (1 \leq i \leq k) && (5) \\ \hat{y}_i &= y_i - \gamma t \quad (1 \leq i \leq k) && (6) \end{aligned}$$

We will assume that $\hat{x}_i \neq 0$ for all i, and hence that $\hat{y}_i \neq 0$ for all i. (See remark 1, below.) $\mathcal{P}$ then secretly generates, randomly and independently from $\mathbf{Z}_q$, $2k-1$ elements, $\theta_1, \ldots \theta_{2k-1}$, computes

$$\begin{array}{rcl} \Theta_1 & = & G^{-\theta_1 \hat{y}_1} \\ \Theta_2 & = & G^{(\theta_1 \hat{x}_2 - \theta_2 \hat{y}_2)} \\ \vdots & = & \vdots \\ \Theta_i & = & G^{(\theta_{i-1} \hat{x}_i - \theta_i \hat{y}_i)} \\ \vdots & = & \vdots \\ \Theta_k & = & G^{(\theta_{k-1} \hat{x}_k - \theta_k \hat{y}_k)} \\ \Theta_{k+1} & = & G^{(\gamma\,\theta_k - \,\theta_{k+1})} \\ \vdots & = & \vdots \\ \Theta_{2k} & = & G^{\gamma\,\theta_{2k-1}} \end{array} \tag{7}$$

and reveals the sequence $\Theta_1, \ldots, \Theta_{2k}$ to $\mathcal{V}$.

SSA. 3. $\mathcal{V}$ generates a random challenge, $c \in \mathbf{Z}_q$ and reveals it to $\mathcal{P}$.

SSA. 4. $\mathcal{P}$ computes $2k-1$ elements, $\alpha_1, \ldots, \alpha_{2k-1}$, of $\mathbf{Z}_q$ satisfying

$$\alpha_i \;=\; \theta_i \;+\; c \prod_{j=1}^{i} \left(\frac{\hat{x}_j}{\hat{y}_j} \right) \qquad 1 \le i \le k \tag{8}$$

$$\alpha_i \;=\; \theta_i \;+\; c\,\gamma^{i-2k} \qquad k+1 \le i \le 2k-1$$

or, equivalently,

$$\alpha_i \;=\; \theta_i \;+\; c\gamma^{-k} \prod_{j=i+1}^{k} \left(\frac{\hat{y}_j}{\hat{x}_j} \right) \qquad 1 \le i \le k-1 \tag{9}$$

$$\alpha_i \;=\; \theta_i \;+\; c\,\gamma^{i-2k} \qquad k \le i \le 2k-1$$

and reveals the sequence α_i to $\mathcal{V}$. (Notice that the α_i can be evaluated recursively using only one $\mathbf{Z}_q$ multiplication and one $\mathbf{Z}_q$ division per index.)

SSA. 5. $\mathcal{V}$ computes

$$U \;=\; G^{-t} \tag{10}$$

$$\begin{array}{rcl} W & = & \Gamma^{-t} \;=\; G^{-\gamma t} \\ \hat{X}_i & = & X_i\,U \quad (1 \le i \le k) \\ \hat{Y}_i & = & Y_i\,W \quad (1 \le i \le k) \end{array} \tag{11}$$

and checks each of the following $2k$ equations:

$$\begin{array}{rcl} \hat{X}_1^{c}\,\hat{Y}_1^{-\alpha_1} & = & \Theta_1 \\ \hat{X}_2^{\alpha_1}\,\hat{Y}_2^{-\alpha_2} & = & \Theta_2 \\ \vdots & = & \vdots \\ \hat{X}_i^{\alpha_{i-1}}\,\hat{Y}_i^{-\alpha_i} & = & \Theta_i \\ \vdots & = & \vdots \\ \hat{X}_k^{\alpha_{k-1}}\,\hat{Y}_k^{-\alpha_k} & = & \Theta_k \\ \Gamma^{\alpha_k}\,G^{-\alpha_{k+1}} & = & \Theta_{k+1} \\ \vdots & = & \vdots \\ \Gamma^{\alpha_{2k-1}}\,G^{-c} & = & \Theta_{2k} \end{array} \tag{12}$$

$\mathcal{V}$ accepts the proof if and only if all of the equations in (12) hold.

(Note that the left hand side of each of the equations in 12 can be evaluated using the technique of simultaneous multiple exponentiation resulting in an effective reduction of the complexity of evaluating all of them from $4k$ exponentiations to roughly $2.8k$ exponentiations. [20])

Before discussing the formal properties of this protocol we observe:

Remark 1 There is a negligible probability (less than k/q) that $\hat{x}_i = 0$ for one or more i (and thus, if $\mathcal{P}$'s original assertion is true, $y_j = 0$ for an equal number of j). Clearly, this event is the same as that of $\mathcal{V}$ simply guessing $\log_G X_i$ without any help from $\mathcal{P}$. In an interactive execution of the problem, the parties might simply remove the offending list elements and restart with a smaller k, since now $\mathcal{V}$ knows an index pair, i, j with $t = \log_G X_i = \log_\Gamma Y_j$. Preferably, in the case of a non-interactive execution, $\mathcal{P}$ should, if possible, "re-frame" the original data in several ways rather than reveal a "lucky guess" to $\mathcal{V}$. One way $\mathcal{P}$ can do this is to pick a random μ and replace G and Γ with G^μ and Γ^μ respectively.

Rather than get overly consumed with details at this point, we simply say that the protocol fails to complete with probability bounded by k/q. For typical values of q ($> 2^{159}$) and any practical value of k ($\ll 2^{80}$), the protocol is complete ([20], 10.18 Definition).

Remark 2 An important part of the problem statement (definition 4 is the "in subgroup assumption" that X_i and Y_i are all elements of $\mathcal{G}$. This assumption is often overlooked because it is something that $\mathcal{V}$ can check without any additional information from $\mathcal{P}$. In [34], Wikström provides some techniques for guaranteeing this condition without need for the usual "explicit" check via exponentiation. The computational cost of explicitly checking is roughly 10% of that of the main protocol, so using Wikström's techniques can provide a noticeable performance improvement. However, specific technique aside, *some method for assuring the "in subgroup assumption" must be used in any actual implementation of the protocol.* This property is not observable from simple inspection.

Remark 3 As has been observed in the case of Chaum-Pedersen, the *size* of a non-interactive proof based on hash function, H, can be significantly reduced by inverting the order of computation. Instead of providing proof transcript $T = (\{\Theta_i\}, \{\alpha_i\})$ and asking $\mathcal{V}$ to compute $c = H(\{X_i\}, \{Y_i\}, \{\Theta_i\})$, $\mathcal{P}$ can provide proof transcript $T' = (c, \{\alpha_i\})$. In this case, $\mathcal{V}$ would compute the Θ_i from equation 7, and then check that $c = H(\{X_i\}, \{Y_i\}, \{\Theta_i\})$.

Theorem 1 The Simple k-Shuffle Proof Protocol satisfies the following properties:

T1.1. It is a four-move, public coin proof of knowledge for the relationship in equation (4).

T1.2. It is complete. Specifically, the probability of completion failure is bounded above by k/q if the condition noted in remark 1 are treated as failures. Otherwise, the protocol always succeeds.

T1.3. The protocol is sound. If $\mathcal{V}$ generates challenges randomly, the *unconditional* (i.e. no restrictions on computation power) probability of a forged proof is less than or equal to

$$(k-1)/q \; + \; 1/q \;\; = \;\; k/q \tag{13}$$

T1.4. It is honest-verifier zero-knowledge.

T1.5. The number of exponentiations required to construct the proof is $2k$, and the number of exponentiations required to verify it is $4k+2$.

Proof:

T1.1 : Obvious.

T1.2 : The proof is contained in remark 1.

T1.3 : A forged proof can *only* be generated in two conditions, both of which are not determined by the computation power of $\mathcal{P}$:

T1.3.1. The challenge t is one of the special values for which

$$\prod_{i=1}^{k}(t - x_i)\,\gamma \;=\; \prod_{i=1}^{k}(\gamma\, t - y_i) \tag{14}$$

that is

$$\gamma^k \prod_{i=1}^{k}(t - x_i) \;=\; \prod_{i=1}^{k}(\gamma\, t - y_i) \tag{15}$$

(Both of these equations are, of course, taken in $\mathbf{Z}_q$.)

T1.3.2. The challenge t is *not* one of the special values in 1 (above), which implies (see [23]) that $(\hat{x}_1, 0, \ldots, 0, -\hat{y}_{2k})$ is not an element of the hyperplane generated by the $2k-1$ vectors

$$\begin{array}{c}
(-\hat{y}_1\,,\, \hat{x}_2\,,\, 0\,,\, \ldots\,,\, 0) \\
(0\,,\, -\hat{y}_2\,,\, \hat{x}_3\,,\, 0\,,\, \ldots\,,\, 0) \\
\vdots \\
(0\,,\, \ldots\,,\, -\hat{y}_{2k-2}\,,\, \hat{x}_{2k-1}\,,\, 0) \\
(0\,,\, \ldots\,,\, 0\,,\, -\hat{y}_{2k-1}\,,\, \hat{x}_{2k})
\end{array} \tag{16}$$

By corollary 2, the probability of T1.3.1 is at most $(k-1)/q$, and the probability of T1.3.2 is $1/q$ by corollary 3. ■

T1.4 : A simulator generates α_i, t and c all randomly and independently (starting over if the negligible probability event, $\hat{X}_i = 1$, occurs). It then determines Θ_i according to equation 12. The resulting distribution is the same as that generated by a real prover and honest verifier. ■

T1.5 : Obvious.

4 Shuffles of ElGamal Pairs

The protocol of this section is essentially the same as that presented in [23], section 6. Some notation has been clarified, and some redundant operations eliminated. A factor of γ in the definition of one set of quantities was also unintentionally omitted in the original. The correction has been added to this presentation.

We briefly restate the problem context in the following definition.

Definition 5 Suppose that two sequences of pairs $(X_1, Y_1), \ldots, (X_k, Y_k)$ (input), and $(\bar{X}_1, \bar{Y}_1), \ldots, (\bar{X}_k, \bar{Y}_k)$ (output), as well as "encryption parameters", g and h, are all publicly known elements of $\mathcal{G} \subset \mathbf{Z}_p^*$. Suppose also, that the prover, $\mathcal{P}$, knows $\beta_1, \ldots, \beta_k$ in $\mathbf{Z}_q$ and $\pi \in \Sigma_k$ such that for all $1 \leq i \leq k$

$$(\bar{X}_i, \bar{Y}_i) \;=\; (g^{\beta_{\pi(i)}} X_{\pi(i)} \;,\; h^{\beta_{\pi(i)}} Y_{\pi(i)}) \tag{17}$$

$\mathcal{P}$ is required to convince $\mathcal{V}$ this fact – that is, convince $\mathcal{V}$ of the existence of β_i and π satisfying equation 17 – without revealing any information about β_i or π.

We call this problem the *ElGamal k-Shuffle Problem.*

4.1 Sketch of the Protocol

Before diving into a detailed presentation of our solution, we will discuss a motivational sketch of the details to follow.

The basic motivation for the protocol is to use lemmas 2 and 3 and their corollaries. To do this, first consider defining, for a fixed permutation, π, β_i and ξ_i by

$$(\bar{X}_i, \bar{Y}_i) \;=\; (g^{\beta_{\pi(i)}} X_{\pi(i)} \;,\; h^{\xi_{\pi(i)}} Y_{\pi(i)}) \tag{18}$$

The goal is then to show that one can find *some* π such that $\beta_i = \xi_i$ for all i. By the referenced lemmas, we can achieve this by showing that for a *random* vector $\vec{s} = (s_1, \ldots, s_k)$,

$$\vec{s} \cdot \vec{\beta} = \vec{s} \cdot \vec{\xi} \tag{19}$$

or, equivalently, for random $\vec{s} = \vec{r}_\pi$ that

$$\vec{s} \cdot \vec{\bar{x}} - \vec{r} \cdot \vec{x} \;=\; \vec{s} \cdot \vec{\bar{y}} - \vec{r} \cdot \vec{y} \tag{20}$$

where we use the notation $\vec{\varepsilon}_\pi = (\varepsilon_{\pi(1)}, \ldots, \varepsilon_{\pi(k)})$.

In order to demonstrate that equation 20 holds while still maintaining the zeroknowledge property of the protocol, the coordinates of $\vec{r}$ and $\vec{s}$ must be kept secret. This forces two steps in the protocol that somewhat obscure the main thrust:

1. So that $\mathcal{V}$ can be sure that the coordinates of $\vec{r}$ are truly *random* without having knowledge of them, $\mathcal{P}$ must first commit a hidden random $\vec{b}$ to which $\mathcal{V}$ can then add a random "offset".
2. So that $\mathcal{V}$ can check equation 20, $\mathcal{P}$ must also create random $\vec{V}$ and $\vec{W}$ that serve the same purpose as obviously similar quantities in the standard Chaum-Pedersen proof. (These quantities are typically named A and B.)

Finally, there is a third subtle issue that must be dealt with. In order to fall back on the linear algebra lemmas we plan to use, one must first fix a "reference permutation," π_0. Without this, $\mathcal{P}$ is free to search *all permutations* for one, π, that satisfies equation 20, and it turns out that for $k! \gg q$, an exhaustive search might prove successful *even if* $\vec{r}$ is completely random. (Nevertheless, it seems that actually computing such a π may be at least as difficult as solving the discrete logarithm problem. This is the thrust of our conjecture 1.) To avoid depending on the conjecture however, we introduce some additional commitments by $\mathcal{P}$ – roughly embodied by the variables $\vec{A}$, $\vec{B}$, $\vec{C}$ and $\vec{D}$. Their reason for existence may seem opaque at first, but should be well clarified by careful inspection of the formal proofs following the presentation of the protocol steps themselves.

4.2 ElGamal k-Shuffle Proof Protocol

EGA. 1. $\mathcal{P}$ generates, for $1 \le i \le k$, u_i, w_i and additionally τ_0 all randomly and independently from $\mathbf{Z}_q$. Further, $\mathcal{P}$ generates a_i uniformly from $\bar{\Delta}_q^{k+}$. (This constraint $\vec{a} \in \bar{\Delta}_q^{k+}$ is not technically required for the success of the protocol. It should become clear shortly that we merely propose it for essentially the same reason that we proposed the constraint $\hat{x}_i \neq 0$ in **SSA.** 2. See remark 4, below.) $\mathcal{P}$ also generates ν, and γ from $\mathbf{Z}_q^*$ randomly and independently. $\mathcal{P}$ then computes

$$\begin{aligned} \Gamma &= g^{\gamma} \\ A_i &= g^{a_i} \\ C_i &= A_{\pi(i)}^{\gamma} = g^{\gamma\, a_{\pi(i)}} \\ U_i &= g^{u_i} \\ W_i &= g^{\gamma w_i} \end{aligned} \tag{21}$$

and

$$\Lambda_1 = g^{\tau_0 + \sum_{i=1}^{k} w_i \beta_{\pi(i)}} \prod_{i=1}^{k} X_i^{w_{\pi^{-1}(i)} - u_i} \tag{22}$$

$$\Lambda_2 = h^{\tau_0 + \sum_{i=1}^{k} w_i \beta_{\pi(i)}} \prod_{i=1}^{k} Y_i^{w_{\pi^{-1}(i)} - u_i} \tag{23}$$

$\mathcal{P}$ then reveals the ordered sequences A_i, C_i, U_i, and W_i along with Γ, Λ_1 and Λ_2 to $\mathcal{V}$.

EGA. 2. For $1 \leq i \leq k$, $\mathcal{V}$ chooses ρ_i randomly and independently from $\mathbf{Z}_q$, computes

$$B_i = g^{\rho_i}/U_i \tag{24}$$

and returns the sequence ρ_i as a challenge to $\mathcal{P}$.

EGA. 3. $\mathcal{P}$ computes

$$b_i = \rho_i - u_i \tag{25}$$

for $1 \leq i \leq k$. (See remark 4, below.) $\mathcal{P}$ then computes

$$\begin{aligned} d_i &= \gamma\, b_{\pi(i)} \\ D_i &= B_{\pi(i)}^{\gamma} = g^{d_i} \end{aligned} \tag{26}$$

and reveals the sequence D_i to $\mathcal{V}$.

EGA. 4. $\mathcal{V}$ generates $\lambda \in \mathbf{Z}_q$ randomly, and returns it to $\mathcal{P}$ as a challenge.

EGA. 5. For $1 \leq i \leq k$, $\mathcal{P}$ secretly computes the exponents

$$\begin{aligned} r_i &= a_i + \lambda\, b_i \\ s_i &= \gamma\, r_{\pi(i)} \end{aligned} \tag{27}$$

(Again, see remark 4, below.) In addition, $\mathcal{P}$ computes for each $1 \leq i \leq k$

$$\sigma_i \;=\; (\gamma\, w_i + d_i)/\gamma \;=\; w_i + \gamma^{-1} d_i \;=\; w_i + b_{\pi(i)} \tag{28}$$

and also

$$\tau \;=\; -\tau_0 + \sum_{i=1}^{k} b_i \beta_i \;=\; -\tau_0 + \sum_{i=1}^{k} b_{\pi(i)} \beta_{\pi(i)} \tag{29}$$

and reveals both τ and the sequence σ_i to $\mathcal{V}$.

EGA. 6. $\mathcal{P}$ and $\mathcal{V}$ then execute the simple k-shuffle, $\mathcal{SS}_k\left(\vec{R},\, \vec{S},\, G,\, \Gamma\right)$, where

$$\begin{aligned} \vec{R} \;&=\; (R_1, \ldots, R_k) \;=\; (g^{r_1}, \ldots, g^{r_k}) \\ \vec{S} \;&=\; (S_1, \ldots, S_k) \;=\; (g^{s_1}, \ldots, g^{s_k}) \end{aligned} \tag{30}$$

(Note that $\mathcal{P}$ need not explicitly compute R_i and S_i in order to construct the proof, while $\mathcal{V}$ can compute R_i and S_i as $R_i = A_i B_i^{\lambda}$ and $S_i = C_i D_i^{\lambda}$.)

EGA. 7. Finally, $\mathcal{V}$ evaluates

$$\Phi_1 \;=\; \prod_{i=1}^{k} \bar{X}_i^{\sigma_i} X_i^{-\rho_i} \tag{31}$$

$$\Phi_2 \;=\; \prod_{i=1}^{k} \bar{Y}_i^{\sigma_i} Y_i^{-\rho_i} \tag{32}$$

and checks that

$$\Gamma^{\sigma_i} \;=\; W_i D_i \tag{33}$$

$$\Lambda_1\, g^{\tau} \;=\; \Phi_1 \tag{34}$$

$$\Lambda_2\, h^{\tau} \;=\; \Phi_2 \tag{35}$$

(Once again, note that the individual terms of the k-fold products in equations 31 and 32 can be evaluated using the technique of simultaneous multiple exponentiation resulting in an effective reduction of the complexity of evaluating each of the two right hand sides from $2k$ exponentiations to roughly $1.4k$ exponentiations. [20])

$\mathcal{V}$ accepts the proof if and only if both of the following:

EGA.7.1. Equations 33, 34 and 35 are all satisfied.

EGA.7.2. $\mathcal{V}$ accepts the simple shuffle proof of step 6.

Remark 4 As was the case with the Simple k-Shuffle protocol (remark 1), there are certain "zero conditions" that we would prefer to protect against when executing the protocol non-interactively. Again, these events are as unlikely as $\mathcal{V}$ simply guessing equivalent information about the input/output data without any help from $\mathcal{P}$, and could just as well be ignored. These conditions are

$$\vec{a} \in \Delta_q^k \cup (\mathbf{Z}_q^k - \mathbf{Z}_q^{*k}) \tag{36}$$
$$\vec{b} \in \Delta_q^k \cup (\mathbf{Z}_q^k - \mathbf{Z}_q^{*k}) \tag{37}$$
$$\vec{\rho} \in \Delta_q^k \cup (\mathbf{Z}_q^k - \mathbf{Z}_q^{*k}) \tag{38}$$

These, of course, imply

$$\vec{c} \in \Delta_q^k \cup (\mathbf{Z}_q^k - \mathbf{Z}_q^{*k}) \tag{39}$$
$$\vec{d} \in \Delta_q^k \cup (\mathbf{Z}_q^k - \mathbf{Z}_q^{*k}) \tag{40}$$
$$\vec{\sigma} \in \Delta_q^k \cup (\mathbf{Z}_q^k - \mathbf{Z}_q^{*k}) \tag{41}$$

Additionally, carrying over the convention of remark 1, there is the condition that step 6 "fails to complete," that is, $\hat{x}_i = 0$ for some i.

As was the case previously, we can adopt the convention that the protocol fails to complete in these cases. Obviously the probability of any of these events occurring is bounded above by the sum of their individual probabilities:

$$2\left((k/q) + k(k-1)/2\right) \; + \; k/q \;\; = \;\; k(k+2)/q \tag{42}$$

Remark 5 Again, an important part of the problem statement (definition 4 is the "in subgroup assumption" that X_i, Y_i, $\bar{X}_i$ and $\bar{Y}_i$ are all elements of $\mathcal{G}$. The warnings of remark 2 are equally important here. However, it is not necessary to do additional "in subgroup verification" for the elements of the sub-protocol in step 6. In the case of a typical mix-net application, it suffices for $\mathcal{V}$ to check the "in subgroup property" only on the output values, $\bar{X}_i$ and $\bar{Y}_i$.

Remark 6 It is worth noting that for very small k, ($k! \ll q$), the protocol can be executed without constructing, A_i or C_i, and simply using $R_i = B_i$ and $S_i = D_i$. In fact, the theorem that follows still holds, but with a forgery probability of roughly $k!/q$ instead of k/q. The reason for this should be obvious from the theorem proof. It does not appear possible to obtain an unconditionally sound protocol without the additional variables when

$k!$ is large, however it may be possible to eliminate them and obtain a computationally sound protocol. Whether or not this is true depends on the following conjecture. Of course, the advantage of eliminating these variables is that the overall exponentiation count is reduced by $4k$.

Conjecture 1 *The following problem is NP hard (perhaps reducible to Subset-Sum):*

Problem 1 Suppose that $\vec{a}\,,\,\vec{b} \in \mathbf{Z}_q^k$, and that for all $\pi \in \Sigma_k$,

$$\vec{a} \neq \vec{b}_\pi \tag{43}$$

For $\vec{e} \in_R \mathbf{Z}_q^k$, find $\sigma \in \Sigma_k$ such that

$$\vec{e} \cdot \vec{a} = \vec{e}_\sigma \cdot \vec{b} \tag{44}$$

or equivalently, such that

$$\sum_{i=1}^{k} e_i a_i = \sum_{i=1}^{k} e_{\sigma(i)} b_i \tag{45}$$

We now return to the analysis of the main protocol.

Theorem 2 The ElGamal k-Shuffle Proof Protocol satisfies the following properties:

T2.1. It is a seven-move, public coin proof of knowledge for the relationship in equation (17).

T2.2. It is complete. Specifically, the probability of completion failure is bounded above by

$$k(k+2)/q$$

if the conditions noted in remark 4 are treated as failures. Otherwise, the protocol always succeeds. (Of course, in a non-interactive implementation, $\mathcal{P}$ is free to "retry" by starting with new values of Γ, etc.)

T2.3. The protocol is sound for $k < q/2$. Specifically, if $\mathcal{V}$ generates challenges randomly, the *unconditional* (i.e. no restrictions on computation power) probability of a forged proof is less than

$$(2k+1)/q + 2/q = (2k+3)/q \tag{46}$$

T2.4. Assuming the Diffie-Hellman Decision Problem is hard, it is computational zero-knowledge.

T2.5. The number of exponentiations required to construct the proof is $8k + 4$, and the number of exponentiations required to verify it is $12k + 4$.

Proof:

T2.1 : It is obvious that the protocol is public coin. As presented, the move count is ten. However, all but the last two moves of the Simple Shuffle execution in step **EGA.**.6 can be executed in parallel with the previous steps of the main protocol.

T2.2 : The proof is contained in remark 4.

T2.3 : We prove this by way of the following lemmas.

Lemma 5 If for more than k values of λ, there is a permutation, π_λ satisfying

$$s_i = \gamma\, r_{\pi_\lambda(i)} \tag{47}$$

then for all $1 \le i \le k$, there exists a j such that

$$\begin{aligned} c_i &= \gamma\, a_j \\ d_i &= \gamma\, b_j \end{aligned} \tag{48}$$

Proof: Fix any one i. Since $\pi_\lambda(i)$ can take on at most k values, there must be two distinct values of λ satisfying 47, λ_1 and λ_2, with $\pi_{\lambda_1}(i) = \pi_{\lambda_2}(i) = j$. From the definition of r_i and s_i (verified by $\mathcal{V}$ in step **EGA.**6) we have

$$\begin{aligned} b_j + \lambda_1\, d_j &= \gamma(a_i + \lambda_1\, c_i) \\ b_j + \lambda_2\, d_j &= \gamma(a_i + \lambda_2\, c_i) \end{aligned} \tag{49}$$

Taking differences of both sides completes the proof. ■

Corollary 5 Under the conditions of lemma 5 there is a *unique* permutation $\pi_0 \in \Sigma_k$ satisfying

$$\begin{aligned} c_i &= a_{\pi_0(i)} && (50) \\ d_i &= b_{\pi_0(i)} && (51) \end{aligned}$$

Proof: Since $a_i \neq a_j$ for $i \neq j$, the conclusion is trivial. ■

Corollary 6 If $\mathcal{P}$ can, with probability greater than

$$k/q \ + \ (k+1)/q \ = \ (2k+1)/q \qquad (52)$$

produce a Simple k-Shuffle transcript which is accepted by $\mathcal{V}$, then the equations in 50 hold.

Proof: If the equations in 50 do not hold, then there are fewer than $k+1$ values of λ for which equations 47 hold. $\mathcal{P}$ can then only succeed in producing a Simple k-Shuffle proof accepted by $\mathcal{V}$ if either of the following two (non-exclusive) conditions hold:

- $\mathcal{V}$ chooses one of the k exceptional values of λ.
- $\mathcal{P}$ creates a forged shuffle proof (accepted by $\mathcal{V}$).

The probability of the latter event, is by the previous section results, bounded above by $(k+1)/q$. The probability of the former is obviously bounded by k/q. Hence the probability of either is bounded by the sum of the individual probabilities. ■

Lemma 6 If the equations in 50 hold, and suppose that for some $1 \leq i \leq k$

$$\log_g(\bar{X}_i / X_{\pi_0(i)}) \ \neq \ \log_h(\bar{Y}_i / Y_{\pi_0(i)})$$

Let P be the (unconditional) probability that $\mathcal{V}$ chooses ρ_i and λ so that both equations 34 and 35 hold. Then

$$P \ < \ \frac{2}{q} \qquad (53)$$

Proof: Define β_i, ϵ_i, λ_1 and λ_2 by

$$\begin{aligned} \beta_{\pi_0(i)} &\doteq \log_g(\bar{X}_i/X_{\pi_0(i)}) && (54)\\ \epsilon_{\pi_0(i)} &\doteq \log_h(\bar{Y}_i/Y_{\pi_0(i)}) - \beta_{\pi_0(i)} && (55)\\ \lambda_1 &= \log_g \Lambda_1 && (56)\\ \lambda_2 &= \log_h \Lambda_2 && (57) \end{aligned}$$

The assumption of the corollary statement is that $\epsilon = (\epsilon_1, \ldots, \epsilon_k) \neq \vec{0}$. Since equations 24 and 33 hold,

If we take $\log_g$ of both sides of equation 34 and $\log_h$ of both sides of equation 35, and invoke equations 31, 32, 24 and 33 we obtain

$$\begin{aligned} \lambda_1 + \tau &= \sum_{i=1}^k b_{\pi_0(i)}\bar{x}_i - \sum_{i=1}^k b_i x_i + K_1 && (58)\\ \lambda_2 + \tau &= \sum_{i=1}^k b_{\pi_0(i)}\bar{y}_i - \sum_{i=1}^k b_i y_i + K_2 && (59) \end{aligned}$$

where K_1 and K_2 are constant over all choices of ρ_i and λ. Using equations 54 and 55 as

$$\begin{aligned} \lambda_1 + \tau &= \sum_{i=1}^k b_{\pi_0(i)}(x_{\pi_0(i)} + \beta_{\pi_0(i)}) - \sum_{i=1}^k b_i x_i + K_1 && (60)\\ \lambda_2 + \tau &= \sum_{i=1}^k b_{\pi_0(i)}(y_{\pi_0(i)} + \beta_{\pi_0(i)} + \epsilon_{\pi_0(i)}) - \sum_{i=1}^k b_i y_i + K_2 && (61) \end{aligned}$$

or, by reordering the first sum on the right hand side of each equation,

$$\begin{aligned} \lambda_1 + \tau &= \sum_{i=1}^k b_i\beta_i + K_1 && (62)\\ \lambda_2 + \tau &= \sum_{i=1}^k b_i(\beta_i + \epsilon_i) + K_2 && (63) \end{aligned}$$

Subtracting equation 62 from equation 63 gives

$$\sum_{i=1}^k b_i\,\epsilon_i = K_0 \qquad (64)$$

where K_0 is a constant over all choices of ρ_i and λ. The conclusion of the lemma thus follows directly from lemma 4. ■

The claims of **T2.**3 follow easily by combining the preceding lemmas and corollaries. ■

T2.4 : A simulated protocol transcript is generated as follows:

T2.4.1. Generate Γ, A_i, ρ_i, B_i and λ as in the actual protocol execution.

T2.4.2. Compute $U_i = g^{\rho_i} B_i^{-1}$.

T2.4.3. Generate σ_i, C_i and D_i randomly, and compute $W_i = \Gamma^{\sigma_i} D_i^{-1}$, subject to the constraints in 39-41.

T2.4.4. Compute $\vec{R}$ as in the actual protocol execution and S_i by $S_i = C_i D_i^{\lambda}$.

T2.4.5. Simulate the Simple k-Shuffle protocol.

T2.4.6. Generate τ randomly from $\mathbf{Z}_q$.

T2.4.7. Compute Λ_1 and Λ_2 from equations 31 and 32 respectively.

Let $\mathcal{S}$ and $\mathcal{S}'$ be the distributions of real and simulated ElGamal shuffle protocol transcripts respectively, with distributions, $\mathcal{D}_\pi$ and $\mathcal{D}_m$ for π and m. We assume that the distribution of encryption exponents, r, on the original list of ElGamal pairs is uniform random. That is the shuffle input, $\{(X_i, Y_i)\}$ satisfies $X_i = g^{r_i}$, where r_i are uniform random. Let $\mathcal{D}$ and $\mathcal{D}'$ be the distributions of real and simulated Diffie-Hellman distributions, that is

$$\begin{aligned} \mathcal{D} &= \{(g^r, h^r)\} \\ \mathcal{D}' &= \{(g^r, h^s)\} \end{aligned} \tag{65}$$

where r and s are random and independent.

The precise statement of **T2.**4 is

Theorem 3 *Suppose that there exists $D_{\mathcal{S}}$ that can distinguish between $\mathcal{S}$ and $\mathcal{S}'$ for some k with probability $1/2 + \epsilon$ and work W. Then $D_{\mathcal{D}}$ can be constructed from $D_{\mathcal{S}}$ that distinguishes between $\mathcal{D}$ and $\mathcal{D}'$ with the same probability, and work $2W$.*

Proof: Let $S_1 = \{(\phi_{1i}, \psi_{1i})\}_{i=1}^k$ and $S_2 = \{(\phi_{2i}, \psi_{2i})\}_{i=1}^k$ be independent sequences of pairs from $\mathcal{D}$ or $\mathcal{D}'$. We construct a ElGamal shuffle transcript from S_1 and S_2. The transcript distribution will be from $\mathcal{S}$ if the pairs are from $\mathcal{D}$, and will be from $\mathcal{S}'$ if they are from $\mathcal{D}'$.

T3.1. Choose π from $\mathcal{D}_\pi$, and m_i from $\mathcal{D}_m$.

T3.2. Generate r_i and β_i randomly as they would be generated in an actual instance of the shuffle protocol. Compute

$$\begin{aligned} X_i &= g^{r_i} \\ Y_i &= h^{r_i} m_i \\ \bar{X}_i &= g^{\beta_{\pi(i)} + r_{\pi(i)}} \\ \bar{Y}_i &= h^{\beta_{\pi(i)} + r_{\pi(i)}} m_{\pi(i)} \end{aligned}$$

T3.3. Generate a random η in $\mathbf{Z}_q^*$, and set $\Gamma = h^\eta$.

T3.4. Generate ϱ_i, and ς_i, $1 \le i \le k$ all randomly and independently, and set

$$\begin{aligned} A_i &= \phi_{1i} \\ B_i &= \phi_{2i} \\ C_i &= \psi_{1\,\pi(i)}^\eta \\ D_i &= \psi_{2\,\pi(i)}^\eta \end{aligned}$$

T3.5. Generate the remaining shuffle transcript variables exactly as generated in the transcript simulation, T2.4.1 - T2.4.7.

The task of establishing the desired properties of the resulting distribution is tedious but straightforward. ■

T2.5 : Obvious.

5 ElGamal Sequences

If the ballot in an election consists of more than one question, it can sometimes be useful to have voters encrypt their “voted ballot” as a *sequence* of ElGamal pairs rather than as a single pair. In this case, the input and output to the shuffle are of the form (X_{ji}, Y_{ji}) and $(\bar{X}_{ji}, \bar{Y}_{ji})$ respectively, $1 \leq i \leq k$, $1 \leq j \leq N_Q$, where k is the number of “cast ballots” and N_Q is the number of “answers” on the ballot. It is required to shuffle the ballots, without changing the order of each message within a given ballot. More formally, the problem is stated as follows:

Definition 6 Suppose that two sequences of sequences of pairs $(X(j,i), Y(j,i))$ (input), and $(\bar{X}(j,i), \bar{Y}(j,i))$ (output), $1 \leq i \leq k$, $1 \leq j \leq N_Q$, as well as “encryption parameters”, g and h, are all publicly known elements of $\mathcal{G} \subset \mathbf{Z}_p^*$. Suppose also, that the prover, $\mathcal{P}$, knows $\beta(j,i)$ in $\mathbf{Z}_q$ and $\pi \in \Sigma_k$ such that for all $1 \leq i \leq k$ and $1 \leq j \leq N_Q$

$$(\bar{X}(j\,,\,i), \bar{Y}(j\,,\,i)) \;=\; (g^{\beta(j\,,\pi(i))}X(j\,,\,\pi(i))\,,\; h^{\beta(j\,,\pi(i))}Y(j\,,\,\pi(i))) \quad (66)$$

$\mathcal{P}$ is required to convince $\mathcal{V}$ this fact – that is, convince $\mathcal{V}$ of the existence of β and π satisfying equation 17 – without revealing any information about β or π.

We call this problem the *ElGamal Sequence Shuffle Problem.*

Since it is specifically required that the ordering with respect to the j index remain constant, this problem can be easily solved by invoking the protocol of the last section:

5.1 ElGamal Sequence Shuffle Proof Protocol

EGAR. 1. $\mathcal{V}$ generates e_j, $1 \leq j \leq N_Q$, randomly and independently, and computes

$$\hat{X}_i \;=\; \prod_{j=1}^{N_Q} X^{e_j}(i\,,\,j) \quad (67)$$

$$\begin{aligned}\hat{Y}_i &= \prod_{j=1}^{N_Q} Y^{e_j}(i\,,\,j)\\ \check{X}_i &= \prod_{j=1}^{N_Q} \bar{X}^{e_j}(i\,,\,j)\\ \check{Y}_i &= \prod_{j=1}^{N_Q} \bar{Y}^{e_j}(i\,,\,j)\end{aligned}$$

and presents the ElGamal k-Shuffle problem $(\hat{X}_i, \hat{Y}_i), (\check{X}_i, \check{Y}_i)$ to $\mathcal{P}$.

EGAR. 2. $\mathcal{P}$ and $\mathcal{V}$ execute an ElGamal k-shuffle proof protocol on this data. For this, $\mathcal{P}$ obviously knows π, and also knows

$$\beta_{\pi(i)} = \sum_{j=1}^{N_Q} e_j \, \beta(j\,,\,\pi(i)) \tag{68}$$

EGAR. 3. $\mathcal{V}$ accepts the sequence shuffle proof, if and only if this ElGamal k-shuffle proof is accepted.

It immediately follows from corollary 4 that the properties of this protocol are nearly identical to those of the ElGamal k-Shuffle protocol on which it is based (theorem 2). The forgery probability bound should be increased by k/q to take into account the possibility that e_j is chosen "badly" for one or more i, and the exponentiation count should be increased by $4kN_Q$.

Remark 7 A careful reader will notice that it suffices to always choose $e_1 = 1$ rather than randomly. By small modification to the counting argument that is the basis for lemma 3, and corollary 4, one can see that the forgery probability remains the same, but the number of additional exponentiations is reduced from $4kN_Q$ to $4k(N_Q - 1)$. This, of course, is to be expected since additional exponentiations should not be required when $N_Q = 1$.

References

[1] M. Abe. *Mix-Networks on Permutation Networks - ASIACRYPT 99*, Lecture Notes in Computer Science, pp. 258-273, Springer-Verlag, 1999.

[2] M. Abe and F. Hoshino. Remarks on Mix-Network Based on Permutation Networks. *Proceedings 4th International Workshop on Practice and Theory in Public Key Cryptography PKC 2001*, Lecture Notes in Computer Science, pages 317-324, Springer-Verlag, 2001.

[3] J. Benaloh. Secret Sharing Homomorphisms: Keeping Shares of a Secret Secret. *Advances in Cryptology - CRYPTO '86*, Lecture Notes in Computer Science, pp. 251-260, Springer-Verlag, Berlin, 1987.

[4] J. Benaloh, M. Yung. Distributing the power of a government to enhance the privacy of voters. *ACM Symposium on Principles of Distributed Computing*, pp. 52-62, 1986.

[5] S. Brands. An Efficient Off-line Electronic Cash System Based On The Representation Problem. CWI Technical Report CS-R9323, 1993.

[6] R. Cramer, I. Damgrd, B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. *Advances in Cryptology - CRYPTO '94*, Lecture Notes in Computer Science, pp. 174-187, Springer-Verlag, Berlin, 1994.

[7] R. Cramer, M. Franklin, B. Schoenmakers, M. Yung. Multi-authority secret-ballot elections with linear work. *Advances in Cryptology - EUROCRYPT '96*, Lecture Notes in Computer Science, Springer-Verlag, Berlin, 1996.

[8] R. Cramer, R. Gennaro, B. Schoenmakers. A secure and optimally efficient multi-authority election scheme. *Advances in Cryptology - EUROCRYPT '97*, Lecture Notes in Computer Science, Springer-Verlag, 1997.

[9] D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84-88, 1981.

[10] D. Chaum. Zero-knowledge undeniable signatures. *Advances in Cryptology - EUROCRYPT '90, Lecture Notes in Computer Science*, volume 473, pages 458-464, Springer-Verlag, 1991.

[11] D. Chaum and T.P. Pedersen. Wallet databases with observers. Advances in Cryptology - CRYPTO '92, volume 740 of *Lecture Notes in Compute Science*, pages 89-105, Berlin, 1993. Springer-Verlag.

[12] A. De Santis, G. Di Crescenzo, G. Persiano and M. Yung. On Monotone Formula Closure of SZK. *FOCS 94*, pp. 454-465.

[13] W. Diffie, M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644-654, 1976.

[14] T. ElGamal. A public-key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, IT-31(4):469-472, 1985.

[15] A. Fujioka, T. Okamoto, K. Ohta. A practical secret voting scheme for large scale elections. *Advances in Cryptology - AUSCRYPT '92*, Lecture Notes in Computer Science, pp. 244-251, Springer-Verlag, 1992.

[16] A. Fiat, A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. *Advances in Cryptology - CRYPTO '86*, Lecture Notes in Computer Science, pp. 186-194, Springer-Verlag, New York, 1987.

[17] J. Furukawa and K. Sako. An Efficient Scheme for Proving a Shuffle. To appear in *CRYPTO 2001.*

[18] R. Gennaro. Achieving independence efficiently and securely. *Proceedings 14th ACM Symposium on Principles of Distributed Computing (PODC '95)*, New York, 1995.

[19] J. Groth. A Verifiable Secret Shuffle of Homomorphic Encryptions. Proceedings of the 6th International Workshop on Theory and Practice in Public Key Cryptography (PKC 2003), pp. 145-160, Springer-Verlag, 2003.

[20] A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone. Handbook of Applied Cryptography, CRC Press, 1997.

[21] I.N. Herstein. *Topics in Algebra.* Weily, 1975.

[22] N. Koblitz, A Course in Number Theory and Cryptography, 2nd edition, Springer, 1994.

[23] C.A. Neff, A Verifiable Secret Shuffle and its Application to E-Voting. *Proceedings ACM-CCS 2001*, 116-125, 2001.

[24] A. M. Odlyzko, Discrete logarithms in finite fields and their cryptographic significance, *Advances in Cryptology - EUROCRYPT '84*, Lecture Notes in Computer Science, Springer-Verlag, 1984.

[25] T. Pedersen. A threshold cryptosystem without a trusted party, *Advances in Cryptology - EUROCRYPT '91*, Lecture Notes in Computer Science, pp. 522-526, Springer-Verlag, 1991.

[26] C. Park, K. Itoh, K. Kurosawa. Efficient anonymous channel and all/nothing election scheme. *Advances in Cryptology - EUROCRYPT '93*, Lecture Notes in Computer Science, pp. 248-259, Springer-Verlag, 1993.

[27] C. P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161-174, 1991.

[28] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612-613, 1979.

[29] K. Sako, J. Kilian. Secure voting using partially compatible homomorphisms, *Advances in Cryptology - CRYPTO '94*, Lecture Notes in Computer Science, Springer-Verlag, 1994.

[30] K. Sako, J. Kilian. Receipt-free mix-type voting scheme – A practical solution to the implementation of a voting booth, *Advances in Cryptology - EUROCRYPT '95*, Lecture Notes in Computer Science, Springer-Verlag, 1995.

[31] J. Kilian, K. Sako, *Secure electronic voting using partially compatible homomorphisms.* U.S. Patent number 5,495,532, filed 8/19/1994, issued 2/27/1996.

[32] J. Kilian, K. Sako, *Secure anonymous message transfer and voting scheme.* U.S. Patent number 5,682,430, filed 1/23/1995, issued 10/28/1997.

[33] S. Vajda. *Patterns and Configurations in Finite Spaces.* Griffin, London, 1967.

[34] D. Wikström. Five Practical Attacks for "Optimistic Mixing for Exit-Polls". SAC 2003.

[35] D. Wikström. Personal communications.