A Latency-Free Election Scheme

Kristian Gjøsteen

October 24, 2006

Abstract

We motivate and describe the problem of finding protocols for multiparty computations that only use a single broadcast round per computation (latency-free computations). We show that solutions exists for one multiparty computation problem, that of elections, and more generally, addition in certain groups. The protocol construction is based on an interesting pseudo-random function family with a novel property.

1 Introduction

Consider a small cluster of stars, separated from each other by distances of between one and four light-years. Being in a civilised part of the universe, the cluster has a general assembly for discussing questions of importance for the cluster. Due to the inconvenience of gathering for this assembly, it has been decided that twice every decade, they shall have a vote on whether to convene the assembly or not.

The communication channel can easily be established with radio telescopes broadcasting a signal to every planet, but there is general agreement about a need for privacy, so some kind of secure election scheme must be used.

Most secure election schemes without a central authority (see for example [2]) that operate over a broadcast channel require two rounds of communication, typically one round to publish encrypted votes and one round to decrypt the result. Unfortunately, due to the speed of light, each extra round will require up to four years to complete, which is clearly too much. Therefore we need an election scheme that can produce a result in a single round.

An election scheme is just one example of a multiparty computation problem. In general, we can consider an environment where a group of users want to perform some type of multiparty computation many times. If they are communicating over high-latency channels, there is a clear incentive to minimise the number of rounds. That raises the natural question: Can we do multiparty computations with just a single round of communication? Of course, some kind of setup will always be necessary, but can we hope to do one computation per additional round, after the setup?

Sometimes it is possible to interleave independent protocol runs, to get essentially one computation result per round. But if we consider situations where the multiparty computations may happen in parallel, interleaving becomes impossible. We propose a solution for one multiparty computation problem (elections) where, after some initial setup, every communication round performs one multiparty group operation, and all communication rounds are essentially independent (any number of new rounds can start before the already started rounds complete).

Franklin and Yung [6] investigated how the communication complexity of multiparty computations could be reduced by performing computations in parallel. This work was in an information-theoretic setting. Our solution does in a certain sense show how to achieve optimal asymptotic communication complexity for certain repeated multiparty group operations: The amount of information broadcast by the user in a single round is equal to the amount of information he inputs into the computation. We emphasise that our results are achieved in the random oracle model (though we discuss how they could be achieved in the common reference string model).

Our main technical contribution in this paper is a very interesting construction for pseudo-random function families with a useful algebraic structure, on which our constructions rely. While the specific construction in Sect. 3.1 has previously appeared in the literature [3] (independent our construction, which first appeared in [7]), the algebraic structure has not been noted before. Theorem 4 is new. These constructions also have interesting applications outside of the current problem domain.

The secondary contribution is the idea that our construction can be used for a certain multiparty computation, specifically elections.

This paper is structured as follows: Sect. 2 contains basic material on pseudo-random function families. Sect. 3 describes two pseudo-random function families with a useful algebraic structure, several useful tools for these function families, and some concrete examples. Sect. 4 discusses requirements for single-round election schemes in general, and describes our proposed election scheme. Finally, in Sect. 5 we make some concluding remarks.

1.1 Notation

For any distribution D and algorithm A, we denote by $A(D)$ the output distribution we get when we sample x from D and run A with input x.

For any set S, $x \stackrel{r}{\leftarrow} S$ denotes that x is sampled from the uniform distribution on S. Following the above notation, we denote by $A(S)$ the output distribution we get when we sample x from the uniform distribution on S and run A with input x.

2 Pseudo-Random Function

Definition 1. Let S_1 and S_2 be sets. A *pseudo-random function family (PRF) F from S_1 to S_2* is a subset of $\mathrm{Map}(S_1, S_2)$ indexed by a key set K: $F = \{f_k : S_1 \to S_2 \mid k \in K\}$.

A *l-distinguisher* A for F is an algorithm that is allowed to query a function chosen either uniformly at random from F or uniformly at random from $\mathrm{Map}(S_1, S_2)$ in at most l points of its choosing, and then output 0 or 1. The advantage of A in distinguishing functions in F from random functions is defined to be

$$\mathrm{Adv}_A = |\Pr[A^f = 1 \mid f \stackrel{r}{\leftarrow} F] - \Pr[A^f = 1 \mid f \stackrel{r}{\leftarrow} \mathrm{Map}(S_1, S_2)]|,$$

where A^f denotes that A is run with oracle access to the function f. We say that F is *(t, ϵ, l)-secure* if no l-distinguisher with advantage at least ϵ and run-time at most t exists.

A *weak l-distinguisher* A for F is an algorithm that is allowed to see a function chosen either uniformly at random from F or uniformly at random from $\mathrm{Map}(S_1, S_2)$ evaluated in l points chosen uniformly at random from S_1, and then output 0 or 1. The advantage of A in distinguishing functions in F from random functions is defined to be

$$\begin{aligned}\mathrm{Adv}_A = |&\Pr[A((x_i, f(x_i))_{i=0}^{L-1}) = 1 \mid x_i \stackrel{r}{\leftarrow} S_1, f \stackrel{r}{\leftarrow} F] \\ &- \Pr[A((x_i, f(x_i))_{i=0}^{L-1}) = 1 \mid x_i \stackrel{r}{\leftarrow} S_1, f \stackrel{r}{\leftarrow} \mathrm{Map}(S_1, S_2)]|.\end{aligned}$$

We say that F is *weakly (t, ϵ, l)-secure* if no weak l-distinguisher with advantage at least ϵ and run-time at most t exists.

In the *random oracle model*, we can construct a secure PRF from S_0 to S_2 using a weakly secure PRF from S_1 to S_2. This result is well-known and we include the proof for completeness.

Theorem 2. *Let F be a pseudo-random function family from S_1 to S_2, let $h : S_0 \to S_1$ be a function chosen uniformly at random from* $\mathrm{Map}(S_1, S_2)$*, and let $F' = \{h \circ f_k \mid f_k \in F\}$. For any l-distinguisher A for F' in the random oracle model making at most L queries to the random oracle h, there exists a weak $L + l$-distinguisher for F with the same advantage.*

Proof. Let A be an l-distinguisher for F' in the random oracle model. This algorithm expects access to two oracles, one for the random function $h : S_0 \to S_1$, and one for the function f to be analysed.

We may assume that before A queries its f oracle with some value, it always queries the h oracle with the same value. We may also assume that all queries to the h oracle are distinct.

The weak $L + l$-distinguisher A' gets as input $(x_i, f_k(x_i))_{i=0}^{L+l-1}$. It then runs A, and will simulate its oracles. Upon the ith query y_i to the h oracle, (y_i, i) is stored in a lookup table and x_i is returned. If y_i is later queried to the f oracle, $f_k(x_i)$ is returned. When A stops and outputs $b \in \{0, 1\}$, A' stops and outputs b.

It is clear that A' simulates the random oracle h perfectly, since the sequence $(x_i)_{i=0}^{L-1}$ is by assumption independent and uniformly distributed on S_1. Likewise, if $f_k \in F$, then $f = h \circ f_k \in F'$, so A' simulates the f oracle perfectly.

The run time of A' is the same as A, except for the time needed to deal with the lookup table for hash queries. This is essentially $O((L+l)\log(L+l))$ work. □

We also need a minor extension of this notion, where we only consider the indistinguishability of part of the function value. Consider a pseudo-random function family F from S_1 to G, where G is a group that has a subgroup J. Consider the set of functions $\tilde{F} = \{x \mapsto f_0(x)f_1(x) \mid f_0 \in F, f_1 \in \mathrm{Map}(S_1, J)\}$. We note that if $J = G$, then $\tilde{F} = \mathrm{Map}(S_1, G)$, and the following notions coincide with the above notions.

An *l-J-distinguisher* for F is an algorithm that is allowed to query a function chosen uniformly at random from either F or $\tilde{F}$ in at most l points of its choosing, and then output 0 or 1. The advantage of the distinguisher is defined as above.

A *weak l-J-distinguisher* for F is an algorithm that is allowed to see a function chosen either uniformly at random from F or uniformly at random from $\tilde{F}$ evaluated in l points chosen uniformly at random from S_1, and then output 0 or 1. The advantage of the distinguisher is defined as above, but can also be expressed as

$$\begin{aligned}\mathrm{Adv}_A = |&\Pr[A((x_i, f(x_i))_{i=0}^{l-1}) = 1 \mid x_i \xleftarrow{r} S_1, f \xleftarrow{r} F] \\ &- \Pr[A((x_i, f(x_i)r_i)_{i=0}^{l-1}) = 1 \mid x_i \xleftarrow{r} S_1, r_i \xleftarrow{r} J, f \xleftarrow{r} F]|.\end{aligned}$$

3 The PRF Construction

We can construct a practical pseudo-random function family as follows: Let G be a cyclic group of order n (which may be prime or composite, known or unknown). Let $F = \mathrm{Hom}(G, G) = \{x \mapsto x^k \mid k \in \mathbb{Z}_n\} \subseteq \mathrm{Map}(G, G)$. The interesting thing about this is that even after we apply the construction in the previous section, the pseudo-random function family F' still has a group structure, namely that of $\mathrm{Hom}(G, G)$. As we shall see, this property is very useful.

When n is not prime, a random element of G may not be a generator. But if n has no small prime factors, the probability $\phi(n)/n$ that an element sampled uniformly at random from G is a generator is very close to 1.

If the group order n is unknown, but we know a reasonable bound on n, say $2^{N-1} < n < 2^N$ for some N, we can still efficiently sample 2^{-t}-close to uniformly from G if we have a generator g, simply by sampling k uniformly from $\{0, 1, \ldots, 2^{N+t} - 1\}$ and computing g^k. The cost of this is at most $2(N + t)$ group operations using a simple square-and-multiply algorithm, compared to $2N$ group operations if the order n is known.

Likewise, we can sample 2^{-t}-uniformly from $\mathrm{Hom}(G, G)$ by sampling uniformly from $\{0, 1, \ldots, 2^{n+t} - 1\}$. Evaluating such a function costs at most $2(N + t)$ group operations, compared to $2N$ group operations if the order n is known.

In the interest of simplicity, we shall in the following ignore the sampling error that comes from sampling not uniformly, but almost uniformly.

3.1 Security Based on DDH

The Decision Diffie-Hellman problem for the group G is to distinguish tuples of the form (g, g^x, g^y, g^{xy}) from tuples of the form (g, g^x, g^y, g^{xy+z}), where g is a generator for the group G and x, y, z are chosen uniformly at random from $\mathbb{Z}_n$. The advantage of a DDH distinguisher A is defined to be

$$\begin{aligned}\mathrm{Adv}_A = |&\Pr[A(g, g^x, g^y, g^{xy}) = 1 \mid x, y \xleftarrow{r} \mathbb{Z}_n] \\ &- \Pr[A(g, g^x, g^y, g^{xy+z}) = 1 \mid x, y, z \xleftarrow{r} \mathbb{Z}_n]|\end{aligned}$$

Any DDH adversary for G with advantage ϵ can trivially be turned into a weak 2-distinguisher for F with advantage ϵ.

Conversely, any weak distinguisher for F can be turned into a DDH distinguisher with essentially the same strength.

Theorem 3. *Let A be a weak L-distinguisher for F with advantage ϵ. Then there exists a DDH distinguisher A' for G with advantage $\epsilon - (1 - \phi(n)/n)$. The run time of A' is the run time of A plus $4L$ exponentiations and L multiplications in the group.*

Proof. The algorithm A' takes the tuple (g, u, v, w) as input and runs A with the input $\{(r_i, s_i)\} = \{(g^{a_i} v^{c_i}, u^{a_i} w^{c_i})\}_{i=1}^{L}$, where a_i, c_i are sampled uniformly at random from $\mathbb{Z}_n$. When A stops and outputs b, A' stops and outputs b.

The work done by A' is running A and generating its input. The latter requires $4L$ exponentiations and L multiplications.

Let x, y, z be such that $(g, u, v, w) = (g, g^x, g^y, g^{xy+z})$.

If $z = 0$, we get that $(r_i, s_i) = (g^{a_i + c_i y}, (g^x)^{a_i + c_i y})$, or $(r_i, f(r_i))$ with $f(r_i) = r_i^x$ and r_i uniformly distributed in G. Then the input given to A exactly matches the case where the function is chosen from F. This means that

$$\begin{aligned}\Pr[A'(g, g^x, g^y, g^{xy}) = 1 \mid x, y \xleftarrow{r} \mathbb{Z}_n] \\ = \Pr[A(\{(r_i, f(r_i))\}) = 1 \mid r_i \xleftarrow{r} G, f \xleftarrow{r} F].\end{aligned}$$

Now suppose $z \neq 0$. If $z \in \mathbb{Z}_n^*$, then

$$(r_i, s_i) = (g^{a_i + c_i y}, (g^x)^{a_i + c_i y} g^{z c_i}).$$

Note that for any pair (r, s), there exists a and c such that $(r, s) = (g^{a+cy}, (g^x)^{a+cy} g^{zc})$ (set $a' = a + cy$ and $b' = x(a + cy) + zc$, and remember that z is invertible modulo n). Therefore the input given to A exactly matches the case where the function is chosen uniformly at random from $\mathrm{Map}(G, G)$. We get that

$$\begin{aligned}\Pr[A'(g, g^x, g^y, g^{xy+z}) = 1 \mid x, y \xleftarrow{r} \mathbb{Z}_n, z \xleftarrow{r} \mathbb{Z}_n^*] \\ = \Pr[A(\{(r_i, f(r_i))\}) = 1 \mid r_i \xleftarrow{r} G, f \xleftarrow{r} \mathrm{Map}(G, G)].\end{aligned}$$

We compute the advantage of A' as

$$\begin{aligned}
\epsilon' &= |\Pr[A'(g, g^x, g^y, g^{xy}) = 1 \mid x, y, z \xleftarrow{r} \mathbb{Z}_n] - \\
&\quad \Pr[A'(g, g^x, g^y, g^{xy+z}) = 1 \mid x, y, z \xleftarrow{r} \mathbb{Z}_n]| \\
&\leq |\Pr[A(\{(r_i, f(r_i))\}) = 1 \mid r_i \xleftarrow{r} G, f \xleftarrow{r} F] - \\
&\quad \Pr[A(\{(r_i, f(r_i))\}) = 1 \mid r_i \xleftarrow{r} G, f \xleftarrow{r} \mathrm{Map}(G, G)]| - (1 - \phi(n)/n) \\
&= \epsilon - (1 - \phi(n)/n),
\end{aligned}$$

which concludes the proof. □

If the group order of G is divisible by small primes, then the above theorem is no longer useful. However, in many cases a useful theorem can be recovered under reasonable assumptions, such as generators being indistinguishable from small powers of generators.

3.2 Security Based on Subgroup Membership

If G has a proper, non-trivial subgroup H, the subgroup membership problem for G and H is to distinguish elements of H from elements of $G \setminus H$. The advantage of a distinguisher A for the subgroup membership problem is defined to be

$$\mathrm{Adv}_A = |\Pr[A(H) = 1] - \Pr[A(G \setminus H)]|.$$

Now suppose G also has a proper, non-trivial subgroup J of order n' such that $J \cap H = \{1\}$ and $G = HJ$. We let $F = \mathrm{Hom}(H, H)$, but consider F to be a pseudo-random function family from the subgroup H to G.

Any subgroup distinguisher with advantage ϵ can trivially be turned into a weak 1-distinguisher for F with advantage $\epsilon(n' - 1)/n'$, since the output of F will always be in the subgroup H. Conversely, we can use a J-distinguisher for F to construct a distinguisher for the subgroup membership problem for G and H.

Theorem 4. *Suppose G has two disjoint subgroups H and J, such that $G = HJ$, and let A be a weak L-J-distinguisher for F with advantage ϵ. Then there exists a distinguisher for the subgroup membership problem for G and H with advantage at least $\epsilon/(2L) - (1 - \phi(n')/n')$. The run time of A' is the run time of A plus at most $L - 1$ samples from H, L samples from J, L exponentiations and multiplications in G.*

Proof. Any function in $\mathrm{Hom}(G,G)$ can be considered a function in F by restriction. If f is sampled uniformly at random from $\mathrm{Hom}(G,G)$, its restriction will be uniformly distributed in F.

Let D_j be the distribution on L pairs induced by the function $(x_i, f(x_i)r_i)$, where x_i are sampled uniformly from H, f is sampled uniformly from F, and r_i is 1 for $j \leq i < L$, but sampled uniformly from J for $0 \leq i < j$.

Suppose A has advantage ϵ in distinguishing between the distribution D_0 and D_L. Then by a standard hybrid argument, A has advantage at least ϵ/L in distinguishing between distributions D_j and D_{j+1} for some j, that is:

$$|\Pr[A(D_j) = 1] - \Pr[A(D_{j+1}) = 1]| \geq \epsilon/L.$$

Let A' be an algorithm that takes as input x in G. It samples x_i, $0 \leq i < L$, $i \neq j$, from H, f from $\mathrm{Hom}(G,G)$, $r'_j, r_{j+1}, \ldots, r_{L-1}$ from J, and a bit b. Then it sets $x_j = x$, $r_j = (r'_j)^b$ and $r_i = 1$, $0 \leq i < j$, and computes the set $\{(x_i, f(x_i)r_i)\}$. It runs A with this input, and A outputs b'. If $b = b'$, A' outputs 1, otherwise 0.

The run time of A' is the run time of A plus $L-1$ samples from H, at most L samples from J, L exponentiations (evaluations of f) and L multiplications.

If the input to A' is sampled from the uniform distribution on H, the input to A is distributed according to D_j if $b = 0$, and D_{j+1} if $b = 1$. We get

$$\begin{aligned}\Pr[A'(H) = 1] &= \frac{1}{2}(\Pr[A(D_j) = 0] + \Pr[A(D_{j+1}) = 1]) \\ &= \frac{1}{2} + \frac{1}{2}(\Pr[A(D_{j+1}) = 1] - \Pr[A(D_j) = 1]).\end{aligned}$$

Next, we consider the case that the input x to A' is sampled from the uniform distribution on $G \setminus H$.

The group G is isomorphic to $G/H \times G/J$. The group $\mathrm{Hom}(G,G)$ is isomorphic to $\mathrm{Hom}(G/H, G/H) \times \mathrm{Hom}(G/J, G/J)$. Let (f_H, f_J) be the image of f in the latter group under the canonical isomorphism. We can determine the distribution of $(x, f(x)(r'_j)^b)$ by looking separately at the distributions in the factor groups G/J and G/H.

We first consider the distribution of the pairs in the group $G/J \times G/J$. Every pair except the jth is computed independently of b, so their distribution must be independent of b. As for $(xJ, (f(x)(r'_j)^b)J)$, since $r'_j \in J$ we have $r'_j J = 1J$, and we get

$$(f(x)(r'_j)^b)J = f_J(xJ)(r'_j J)^b = f_J(xJ).$$

The distribution of the pairs in $G/J \times G/J$ is therefore independent of b.

Next, we consider the distribution of $(xH, (f(x)(r'_j)^b)H)$ in $G/H \times G/H$. We get

$$(f(x)(r'_j)^b)H = f_H(xH)(r'_jH)^b.$$

For $b = 1$ this is uniformly distributed in G/H and independent of x, since r'_j is uniformly distributed in J.

If $b = 0$, we must compute the distribution of $f_H(xH)$. Since for all pairs (x_i, z_i), $i \neq j$, $(x_i, z_i) \in H \times H$, we have that $(x_iH, z_iH) = (1H, 1H)$ and therefore independent of f_H. If xH is a generator for G/H, $f_H(xH)$ will be uniformly distributed over G/H, and independent of all the other pairs and x.

We conclude that the distribution of $(x_i, f(x_i)r_i)$ is independent of b when $xH = x_jH$ is a generator for G/H. We get

$$|\Pr[A'(G \setminus H) = 1] - \frac{1}{2}| \leq 1 - \phi(n')/n'.$$

Summing up, we get that

$$\begin{aligned}
\epsilon' &= |\Pr[A'(H) = 1] - \Pr[A'(G \setminus H) = 1]| \\
&\geq |\frac{1}{2} + \frac{1}{2}(\Pr[A(D_{j+1}) = 1] - \Pr[A(D_j) = 1]) - \frac{1}{2}| - (1 - \phi(n')/n') \\
&= \frac{1}{2}\epsilon/L - (1 - \phi(n')/n'),
\end{aligned}$$

which concludes the proof. □

We remark that if we are willing to accept that both DDH and the Subgroup Membership problems are hard, we can get tighter bounds in the security proof for the family F. However, the bounds we have established are sufficient for our uses.

3.3 Useful Zero-Knowledge Proofs

In an election scheme, it is vital that every voter proves the correctness of his vote. To do this and still preserve zero latency, we must use non-interactive zero-knowledge proofs. Since we already employ the random oracle model, we can use standard honest-verifier zero-knowledge proofs since in the random oracle model, these can be converted to non-interactive zero-knowledge proofs.

The zero knowledge proofs in this section are all completely standard, and we skip the proofs for completeness and honest-verifier zero-knowledge. Soundness is proved in Sect. 3.4

We specify two parameters, t and N. The security parameter t determines how easy it is for a cheating prover to convince the verifier, it is chosen so that the probability 2^{-t} is sufficiently low. If the group order n is known, $N = n$. Otherwise, N is chosen so that the uniform distributions on $\{0, 1, \ldots, N-1\}$ and $\{ae, ae+1, \ldots, N+ae-1\}$ are statistically close for any $0 \leq a < n$ and $0 \leq e < 2^t$, say $N \approx 2^{2t}n$.

3.3.1 Correct Evaluation

The first proof is that a we have correctly evaluated $f \in F$, relative to a known function value, or alternatively, of equality of discrete logarithms. The prover P wants to prove that there exists a such that $h_0 = g_0^a$ and $h_1 = g_1^a$, for some g_0 and g_1. This amounts to showing that $(h_0, h_1) = (g_0, g_1)^a$, and we can do that by proving that we know a logarithm of (h_0, h_1) to the base (g_0, g_1).

The prover's private input is a, the public input is (g_0, g_1), (h_0, h_1) such that $(h_0, h_1) = (g_0, g_1)^a$.

1. The prover chooses x uniformly at random from $\mathbb{Z}_N$, computes $(z_0, z_1) = (g_0, g_1)^x$, and sends (z_0, z_1) to the verifier..

2. The verifier chooses e uniformly at random from $\{0, 1, \ldots, 2^t - 1\}$ and sends e to the prover.

3. The prover computes $y = x + ea$ and sends y to the verifier.

The verifier accepts if $(g_0, g_1)^y = (z_0, z_1)(h_0, h_1)^e$.

We note that given an e, we can produce an accepting conversation (z_0, z_1, e, y) by choosing y uniformly at random from $\{0, 1, \ldots, N-1\}$ and setting $(z_0, z_1) = (g_0, g_1)^y (h_0, h_1)^{-e}$.

Theorem 5. *The above protocol is complete and honest verifier zero knowledge.*

3.3.2 One of Two is Correct

The second proof is that for a given $f \in F$, one out of two values correspond to the correct value of $f(x)$ for some x. Again, for our family F this corresponds to showing that one out of two pairs have the same discrete logarithm as a reference pair. We prove this by running the previous proof in parallell and tying the two runs together through the challenge. The prover fakes an accepting conversation for the incorrect value and then creates an accepting conversation for the correct.

The prover's private input is a and b, the public input is (g_0, g_1), (h_{00}, h_{01}), (h_{10}, h_{11}) such that $(h_{b0}, h_{b1}) = (g_0, g_1)^a$.

1. The prover generates an accepting conversation $(z_{1-b,0}, z_{1-b,1}, s_{1-b}, y_{1-b})$ for $(h_{1-b,0}, h_{1-b,1})$ by choosing s_{1-b} uniformly at random from $\{0, 1, \ldots, 2^t - 1\}$, y_{1-b} uniformly at random from $\{0, 1, \ldots, N-1\}$ and computing $(z_{1-b,0}, z_{1-b,1}) = (g_0, g_1)^{y_{1-b}}(h_{1-b,0}, h_{1-b,1})^{-s_{1-b}}$.

   He then chooses x uniformly at random from $\{0, 1 \ldots, N-1\}$ and computes $(z_{b,0}, z_{b,1}) = (g_0, g_1)^x$.

   The prover then sends (z_{00}, z_{01}) and (z_{10}, z_{11}) to the verifier.

2. The verifier chooses e uniformly at random from $\{0, 1, \ldots, 2^t - 1\}$ and sends e to the prover.

3. The prover chooses s_b from $\{0, 1, \ldots, 2^t - 1\}$ such that $s_0 + s_1 \equiv e \mod 2^t$ and computes $y_b = as_b + x$ and sends s_0, s_1, y_0, y_1 to the verifier.

The verifier accepts if $s_0 + s_1 \equiv e \pmod{2^t}$ and $(g_0, g_1)^{y_i} = (z_{i,0}, z_{i,1})(h_{i,0}, h_{i,1})^e$ for $i = 0, 1$.

Theorem 6. *The above protocol is complete and honest verifier zero knowledge.*

This protocol can obviously be extended to one out of k by running k proofs in parallel, faking conversations for $k-1$ of them and creating the correct proof for the final one.

3.4 Group Structures

3.4.1 Prime Ordered Groups

The standard group structure for this construction is a group G of known prime order, say the group of rational points on an elliptic curve, or the multiplicative subgroup of a finite field. If the group itself is not of prime order, we can take G to be any prime-ordered subgroup such that the cofactor is relatively prime to the subgroup order. Typically, under Decision Diffie-Hellman, we know that computing discrete logarithms must be hard, and then we get soundness for the protocols in Sect. 3.3.

Theorem 7. *Suppose P^* is a cheating prover for either of the two protocols in Sect. 3.3 instantiated with a group G that succeeds with probability ϵ. If the group order n is known and larger than 2^t, then there exists an algorithm that computes discrete logarithms in G. This algorithm has success probability $\epsilon(\epsilon - 2^{-t+1})$ and run time at most twice that of P^*.*

Proof. We can turn any discrete logarithm problem (g_0, h_0) into the public input simply by choosing any random x and setting $g_1 = g_0^x$, $h_1 = h_0^x$.

Let (z_0, z_1, e, y) and (z_0, z_1, e', y') be two accepting conversations for the protocol from Sect. 3.3.1 with $e \neq e'$. We can by rewinding produce two such conversations with probability at least $\epsilon(\epsilon - 2^{-t+1})$, and the total run time is not more than twice one run of P^*. Observe that $z_0 = g_0^y h_0^{-e}$ and $z_0 = g_0^{y'} h_0^{-e'}$.

Since n is prime and larger than 2^t, we know that $e - e'$ is invertible modulo n we get

$$h_0 = g_0^{(y-y')/(e-e')},$$

and this completes the proof. □

3.4.2 Paillier's Group

Another useful structure, especially for election schemes, is $\mathbb{Z}^*_{n^{s+1}}$ where n is a product of two prime numbers such that n is relatively prime to the order of $\mathbb{Z}^*_n$. As first described by Paillier [9] and elaborated on by Damgård and Jurik [2], $\mathbb{Z}^*_{n^{s+1}}$ contains a subgroup isomorphic to $\mathbb{Z}^*_n$ that is plausibly hard to distinguish (this would be our H), and a subgroup of order n^s where discrete logarithm computations are easy (this would be our J). This subgroup membership problem is know as the Decision Composite Residuosity problem.

Note that we have a nice map from $\mathbb{Z}^*_n$ into $\mathbb{Z}^*_{n^{s+1}}$ given by taking any representative r for the residue class x and taking it to the residue class y with representative r^{n^s}.

We shall assume that n is a product of two safe primes and consider only the quadratic residues, since this simplifies arguments. (This is not essential, however.) The most natural construction to apply in this situation is that of Theorem 4 (note that there are no small primes in the order of J). However, if we do not trust the hardness of the DCR problem, we could use Theorem 3 and rely on Decision Diffie-Hellman, at a modest computational cost.

If it is possible to find a multiple of the order of $\mathbb{Z}^*_n$, then we can easily factor the modulus n. This would allow us to solve the DCR problem, since we can recover a multiple of the group order of H. It would also allow us to apply Theorem 7. So the following theorem is sufficient to prove soundness for the zero-knowledge protocols in Sect. 3.3 under the hardness of either DDH or DCR problems.

Theorem 8. *Suppose P^* is a cheating prover for either of the two protocols in Sect. 3.3 instantiated with $\mathbb{Z}_{n^{s+1}}^*$ that succeeds with probability ϵ. Then there exists an algorithm that computes a multiple of the group order of $\mathbb{Z}_n^*$ with success probability essentially $\epsilon(\epsilon - 2^{-t})$ and run time at most twice that of P^*.*

Proof. We consider first the protocol in Sect. 3.3.1. Let (z_0, z_1, e, y) and (z_0, z_1, e', y') be two accepting conversations with $e \neq e'$. We can by rewinding produce two such conversations with probability at least $\epsilon(\epsilon - 2^{-t+1})$, and the total run time is not more than twice one run of P^*. Observe that $z_0 = g_0^y h_0^{-e}$ and $z_0 = g_0^{y'} h_0^{-e'}$.

Let N be the parameter used in the proof. Choose g_0' uniformly at random by computing the square of a random element in $\mathbb{Z}_n^*$. If g_0' is not a generator of the quadratic residues, we have factored n and we can easily output the group order $\mathbb{Z}_n^*$. Otherwise, let g_0 be the corresponding generator for H (found by computing the n^s power).

Now choose random $N < x < 2N$, and set $h_0 = g_0^x$. From our two accepting conversations, we get

$$g_0^y h_0^{-e} = g_0^{y'} h_0^{-e'} = 1 \quad \text{or} \quad g_0^{x(e'-e)+(y-y')} = 1.$$

Since $y - y' < N$, the exponent is not equal to 0, so $x(e - e') + (y - y')$ is a multiple of the order of g_0, and if multiplied by 4, a multiple of the order of $\mathbb{Z}_n^*$.

To conclude the proof, we observe that for the protocol in Sect. 3.3.2, if we have two accepting conversations for two distinct challenges, then we must have at least one pair of accepting conversations for the first protocol, and by the above arguments we are done. □

4 The Election Scheme

We assume that there is a broadcast channel available. We want an election scheme that can be used for multiple sequential or parallell elections after some initial setup, and that satisfies the following functional requirements:

Constant-Round Setup The number of rounds in the setup phase must be independent of the number of voters and the number of elections that are to be held.

Single Round per Election Each of the multiple elections must require just one round. No voter must be required to decide on his vote before the start of the round, and after that round, every voter must be in possession of the result for that election round, unless some fault occurred.

Any election scheme must satisfy at least the following security requirements:

Privacy Every vote must be as secret as possible in an election (e.g. if the result indicates that all votes were equal, no vote can possibly be private, regardless of the system).

Correctness No voter should be able to submit incorrect votes.

Verifiability Every voter should be able to verify that the tallying was performed correctly.

Usually, election schemes are also required to be *robust*, in the sense that a few voters cannot prevent the remaining voters from computing the result. Schemes that satisfy our functional requirements cannot be robust in this sense. If the scheme allows voters to compute the correct result when one or more votes are missing, any voter could first compute the correct result for all the votes, then pretend that vote i is missing and compute this result. That would reveal the ith vote, and privacy would be lost.

One might relax the functional requirements and say that the single round requirement should hold except in the presence of faults, when a fall-back election protocol should be used to compute the result. Unfortunately, this would allow an inside attacker that can read the ith vote, but prevent it from being broadcast to the other voters, to break the privacy of the ith voter. He could compute the complete result on his own, and the result without the ith vote together with the other voters.

It seems therefore that our functional election requirements forces us to accept schemes that are somewhat fragile in the presence of faults. One possible approach would be to simply re-run the election until no fault occur. If a voter is consistently faulty, that voter could be removed from the voter roll. Note that a denial of service attack is always possible, and against any election protocol, if the attacker controls the entire network.

4.1 The Basic Scheme

We now describe our proposed election scheme, which takes the form of a *yes* or *no* election, encoded as 1 and 0, respectively. The t voters want to execute at most L elections, sequentially or partially in parallell. To focus on the interesting part, we assume that we have a trusted dealer available for now.

We assume that we have the tools developed in the previous section: A pseudo-random function family F' from the set $\{0, 1, \ldots, L\}$ into the group G with a group structure on F', along with a one-out-of-two proof for F'.

Dealer The dealer chooses an element g from G (either a generator for G or for a subgroup J), and for each user a function f_i uniformly at random from F'. He computes the function $f = f_1 f_2 \cdots f_t$. Then he sends f_i privately to the ith user, $i = 1, 2, \ldots, t$, and broadcasts $(g, f, f_1(0), f_2(0), \ldots, f_t(0))$ to every user.

Vote creation In the jth election, voter i encrypts his vote $v_{i,j} \in \{0, 1\}$ as follows: First he computes $c_{i,j} = f_i(j) g^{v_{i,j}}$. He creates a one-out-of-two proof $p_{i,j}$ that proves that one of the values $c_{i,j}$ or $c_{i,j}/g$ is the correct value for $f_i(j)$. Then he broadcasts $(c_{i,j}, p_{i,j})$ to every user.

Tallying The ith voter has the votes $\{c_{l,j}\}_l$ for the jth election, along the the proofs $\{p_{l,j}\}_l$. He verifies the one-out-of-two proofs (stopping if any proof fails), computes $r_j = (\prod_l c_{l,j})/f(j)$, and then computes the result v_j by computing the discrete logarithm of r_j to the base g.

Note that computing the vote count will always be feasible, since the number of votes is at most t.

Privacy This scheme preserves the privacy of every vote because F' is a pseudo-random function family. Giving away the product function f does not compromise this.

Correctness The non-interactive zero-knowledge proof ensures that every vote is correct. Since every other action is performed either by the trusted dealer or the voter himself, this is sufficient to ensure correctness.

Verifiability Again, since the votes are verified to be correct and every other action is performed either by the trusted dealer or the voter himself, every voter will know that the result is correct if the tallying procedure completes.

If we need to run something more complicated than a yes or no election, we could encode votes and use proofs as described in [5], although this would most likely require the Paillier-based [9] group structure, otherwise computing the discrete logarithm would be too expensive.

4.2 Removing the Dealer

We would like to remove the trusted dealer from the scheme. The choice of the element g used to encode the votes as group elements is arbitrary, since the pseudo-random function family hides any value in the subgroup generated by g equally well.

All that remains is for each player to choose f_i and to compute a joint representation for f without each player revealing their secret function, nor allowing any player to cheat. The solution depends on whether the group order is known or unknown.

4.2.1 Known prime group order

When the group order n is known and prime, everything is simple and we do essentially a verifiable multiparty addition. At the start, every voter chooses their function f_i simply by choosing an exponent a_i uniformly at random from $\{0, 1, \ldots, n-1\}$.

In the first round, every voter sends a share of a_i secretly to every other voter. He also commits to his choice by broadcasting $f_i(0)$ along with a non-interactive zero-knowledge proof of knowledge of possession of the key. (One possibility is essentially the proof given in Sect. 3.3.1.)

Then every voter verifies every non-interactive zero knowledge proof, adds every secret key share he has received, and in the second round publishes the sum of all the shares. Finally, every voter adds together all the share sums, to get the number $a = \sum_i a_i$, and this number defines $f = \prod_i f_i$. Note that the correctness of this result can be verified by computing $f(0)$.

4.2.2 Modulo a Power of an RSA Modulus

Now we consider the case of $\mathbb{Z}^*_{n^{s+1}}$. Note that the modulus can be jointly generated using for example the protocol in [1], which supposedly can be made robust against cheating.

Let N be as in Sect 3.3, and set $Q = 2tN$. Every voter chooses his function f_i by choosing an exponent a_i uniformly at random from $\{0, 1, \ldots, N-1\}$. Note that f_i is sampled almost uniformly at random. The idea is now to add the exponents together using multiparty addition modulo Q.

In the first round, every voter sends a share of a_i secretly to every other voter. He also commits to his choice by broadcasting $f_i(0)$ along with a non-interactive zero-knowledge proof of knowledge of possession of the key. (One possibility is essentially the proof given in Sect. 3.3.1, except that since the known exponent a_i is very large, we must use even larger numbers in the proof to hide a_i.)

Then every voter verifies every non-interactive zero knowledge proof, adds every secret key share he has received, and in the second round publishes the sum of all the shares. Finally, every voter adds together all the share sums, to get the number $a = \sum_i a_i$ (which is the integer sum), and this number defines $f = \prod_i f_i$. Note that the correctness of this result can be verified by computing $f(0)$.

5 Concluding Remarks

We have described and motivated the problem of doing multiparty computations in a single communication round. We have also shown that this is possible, by giving an election scheme.

If we skip the vote verification parts of the election scheme, we are left with a general multiparty group operation protocol that is secure in the honest-but-curious model. With the construction from Sect. 3.1, this protocol achieves asymptotically optimal broadcast communication complexity: The user contributes one group element to the multiparty computation and broadcasts one group element per computation. It is impossible to achieve lower broadcast communication complexity.

Our election scheme is realised in the random oracle model. While this is a very good heuristic for security in the real world, many people would prefer schemes in some weaker cryptographic model. It is possible to realise our scheme in the common reference string model, where the common reference string replaces the random values derived from the random function in the construction of the PRF. Obviously, the size of the reference string will limit the number of possible rounds. The non-interactive zero-knowledge proofs would have to be replaced with proofs that work in the common reference string model (see for example [8], or [4] for a somewhat different model).

We have not yet considered the general problem of what kind of multiparty computations can at all be performed in a single broadcast round. This is currently an open problem.

Acknowledgements

Thanks to Susanna tom Raad for posing the problem that lead to the construction in Sect. 3, and to David Wagner and Ivan Damgård for helpful discussions.

References

[1] Joy Algesheimer, Jan Camenisch, and Victor Shoup. Efficient computation modulo a shared secret with application to the generation of shared safe-prime products. In Moti Yung, editor, *CRYPTO*, volume 2442 of *LNCS*, pages 417–432. Springer-Verlag, 2002.

[2] I. Damgård and M. Jurik. A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In Kwangjo Kim, editor, *Proceedings of Public Key Cryptography 2001*, volume 1992 of *LNCS*, pages 119–136. Springer-Verlag, 2001.

[3] Ivan Damgård, Kasper Dupont, and Michael Østergaard Pedersen. Unclonable group identification. In Vaudenay [10], pages 555–572.

[4] Ivan Damgård, Nelly Fazio, and Antonio Nicolosi. Non-interactive zero-knowledge from homomorphic encryption. In Shai Halevi and Tal Rabin, editors, *TCC*, volume 3876 of *Lecture Notes in Computer Science*, pages 41–59. Springer, 2006.

[5] Ivan Damgård, Jens Groth, and Gorm Salomonsen. The theory and implementation of an electronic voting system. In D. Gritzalis, editor, *Secure Electronic Voting*. Kluwer Academic Publishers, 2002.

[6] Matthew Franklin and Moti Yung. Communication complexity of secure computation. In *Proceedings of the 24th ACM STOC*, 1992.

[7] Kristian Gjøsteen. Re: Conditional decryption. Posted to USENET `sci.crypt`, message-id `d550oi$922$1@orkan.itea.ntnu.no`, May 2005.

[8] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Perfect non-interactive zero knowledge for np. In Vaudenay [10], pages 339–358.

[9] P. Paillier. Public-key cryptosystems based on composite degree residue classes. In Jacques Stern, editor, *Proceedings of EUROCRYPT '99*, volume 1592 of *LNCS*, pages 223–238. Springer-Verlag, 1999.

[10] Serge Vaudenay, editor. *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006, Proceedings*, volume 4004 of *Lecture Notes in Computer Science*. Springer, 2006.