Yale University
Department of Computer Science

# Verifiable Secret-Ballot Elections

Josh Daniel Cohen Benaloh

YALEU/DCS/TR-561
September 1987

This work was supported in part by the National Science Foundation under grants MCS-8116678, MCS-8305382, and DCR-8405478 and by the National Security Agency under grant MDA904-84-H-0004.

Abstract

# Verifiable Secret-Ballot Elections

Josh Daniel Cohen Benaloh

Yale University

1987


Privacy in secret-ballot elections has traditionally been attained by using a ballot box or voting booth to disassociate voters from ballots. Although such a system might achieve privacy, there is often little confidence in the accuracy of the announced tally. This thesis describes a practical scheme for conducting secret-ballot elections in which the outcome of an election is verifiable by all participants and even by non-participating observers. All communications are public, yet under a suitable number-theoretic assumption, the privacy of votes remains intact.

The tools developed here to conduct such elections have additional independent applications. *Cryptographic capsules* allow a prover to convince verifiers that either statement $A$ or statement $B$ is true without revealing substantial information as to which. *Secret sharing homomorphisms* enable computation on shared (secret) data and give a method of distributing shares of a secret such that each shareholder can verify the validity of all shares.

# Verifiable Secret-Ballot Elections

A Dissertation
Presented to the Faculty of the Graduate School
of
Yale University
in Candidacy for the Degree of
Doctor of Philosophy

by
Josh Daniel Cohen Benaloh
December 1987

to my wife LAURIE
in the spirit of BENALOH

# Acknowledgements

I owe a great many debts to people who have given me generous help with much of this work.

My advisor, Mike Fischer, has had the often quite difficult task of teaching me the discipline required to do work in theoretical computer science. He has used his marvelous intuition to channel my efforts away from the many dead ends which I would have happily spent months, if not years, exploring. Mike has also taught me the importance of the hard work that, although unpleasant, is necessary to make clever ideas into worthwhile results.

Shafi Goldwasser and Neil Immerman served as my other official readers. Neil read my work with tremendous patience and care. He has more than once given his time and talents to help me over trouble spots, and his great enthusiasm has often added enjoyment to what would otherwise be the dullest of tasks. Shafi has shown great interest in my work from its earliest forms. She has contributed much to the field, and my work simply would not have been possible without the fundamental achievements of which she is one of the leading pioneers.

There are many others to whom I am indebted. Silvio Micali has been enthusiastic about my ideas in the many conversations we have had, and this work is built upon many of the important ideas which he has contributed. Oded Goldreich has been very supportive of my work, and he added a key insight which led to one of the major applications. Dana Angluin has a wonderfully vast knowledge of theoretical computer science which she has always shared generously along with her keen perspective of the field and of life in general. Lenny Pitt and Gregory Sullivan served as beacons to show that there is indeed life after Yale. Dave Greenberg, Ruben Michel, and David Wittenberg have always been willing to listen to my ideas, add their own, and generally make life as a graduate student far more pleasant. Among the others who have been of great help to me are Gilles Brassard, Don Coppersmith, Claude Crépeau, Paul Feldman, Joe Kilian, Jerry Leichter, Moti Yung, and many more to whom I now apologize for having omitted.

Most of all, I want to express my deepest gratitude to my wife Laurie. Besides her constant support and her assumption of many chores which I neglected during the more intensive periods of this work, she took the time to carefully

read the first draft of this thesis. She found typographical errors, suggested clarifications, and simplified proofs. Her drudge work enabled my other readers to receive a far more polished product, and if I'd incorporated more of her recommendations when they were made, I would have had fewer complaints about later versions from my other readers.

# Contents

# Chapter 1

# Introduction and Overview

Over the last decade, the field of public-key cryptography has flourished within theoretical computer science. Techniques based upon infeasible computations now exist which allow one to perform tasks that seem to defy intuition.

This thesis broadens the scope of some known techniques and develops a number of new ones. Some of the methods given here also greatly simplify many previous cryptographic protocols.

Although there are many applications for the tools presented here, there is one common thread throughout this work. Virtually all of the tools described are incorporated to construct a schema for holding secret-ballot elections in which the outcome of an election is verifiable by all participants and observers. This schema has a number of desirable properties.

- The schema is practical. The necessary protocols can be run in reasonable time with current technology. In addition, the protocols contain a large amount of intrinsic parallelism. Expected future technologies can make these protocols extremely fast.

- Every participant (and even mere observers) can verify the tally of an election. They can be convinced that, with extremely high probability, the election tally given represents the true tally of the election.

- The schema is robust. No action can be taken by any voter or set of voters which will corrupt or disrupt the election in any way. Failure of up to half of the appointed "tellers" (or vote counters) can be withstood without corrupting or disrupting the election.

- The secrecy of legitimate votes remains intact (under a cryptographic assumption to be described later) even if up to half of the tellers and an arbitrary set of voters collude. This number can be increased up to the point where if even one teller remains honest, then privacy of votes is maintained, but there is a corresponding decrease in the number of teller failures which an election can survive.

One of the additional applications developed here arises from the notion of *secret sharing homomorphisms.* Secret sharing (as developed by Blakley and Shamir) allows shares of a secret to be distributed to many shareholders in such a way that any subset of shareholders of size beyond a predetermined threshold can reconstruct the secret. Smaller subsets, however, can obtain no information whatsoever about the secret.

It is shown that in many secret sharing schemes, computations on a secret or set of secrets can be accomplished by computing directly upon the corresponding shares. Thus, for a wide variety of simple arithmetic functions, shares of the result of applying the function to the secret(s) can be obtained by applying related functions directly to shares of the secret(s).

When secret sharing homomorphisms are combined with certain encryption methods described in this work, some powerful new tools can be created. One of these tools gives a practical mechanism whereby the holder of a secret can distribute shares to a number of agents in such a way that each agent can be convinced that it holds a legitimate share of the secret. This task, called *verifiable secret sharing*, has been studied in existing literature, but this is the first practical construction of a verifiable secret sharing scheme.

The notion of *cryptographic capsules* is another tool which has applications beyond elections. Capsules provide a simple mechanism which allows one agent to convince a second that one of two (or more) statements is true without revealing which is the case. This tool is useful in a wide variety of interactive protocols, some of which will be described.

### Structure of this Thesis

Chapter 2 develops the theory of higher residues and an encryption mechanism based on this theory. The chapter concludes with an oversimplified first pass at verifiable secret-ballot elections.

Chapter 3 gives an overview of *interactive proofs* and introduces the notion

of *cryptographic capsules*. It is shown how capsules can be used to enhance the capabilities and simplify many uses of interactive proofs. This chapter concludes with a second pass at elections which shows how to use interactive proofs and cryptographic capsules to plug some of the gaps left by the first pass.

Chapter 4 describes the concept of *secret sharing* (or *threshold*) schemes and the rather surprising homomorphism properties which allow direct computations on shares to be performed in many such schemes. As an application of secret sharing homomorphisms, this chapter describes a simple and practical method of verifiable secret sharing. Those already familiar with interactive proofs can read this chapter independently of chapter 3.

Chapter 5 ties together the ideas of the previous three chapters by using them to construct a general schema for practical, robust, and (most importantly) verifiable secret-ballot elections. The properties claimed are carefully delineated, and rigorous proofs are given that these properties are actually attained.

# Chapter 2

# The Election Encryption Function and its Properties

Since 1976, when Diffie and Hellman first published their landmark paper *New Directions in Cryptography* ([DiHe76]), public-key cryptography has blossomed into a major discipline within theoretical computer science. In 1978, Rivest, Shamir, and Adleman published the first true public-key cryptosystem ([RSA78]). Their system was based upon Euler's theorem and the security of their scheme relies upon the difficulty of factoring large integers. From that time on, number-theoretic methods have been closely associated with public-key cryptography.

In order to proceed, a small amount of background and notation is necessary.

**Definition** We say that $a \mid b$ (read "$a$ divides $b$") if and only if there exist an integer $m$ such that $am = b$. We write $a \nmid b$ to indicate that it is not the case that $a$ divides $b$.

**Definition** We say that $a \equiv_n b$ (read "$a$ is equivalent to $b$ modulo $n$") if and only if $n \mid (a - b)$.

**Definition** We use $a \bmod n$ to denote the unique integer $b$ such that $0 \leq b < n$ and $a \equiv_n b$.

**Definition** Let $\mathbf{Z}_n^* = \{\text{integers } x : 0 < x < n, \gcd(x, n) = 1\}$ denote the multiplicative subgroup of integers modulo $n$.

**Definition** Denote by $\varphi(n)$ the size of the group $\mathbf{Z}_n^*$. ($\varphi(N)$ is often called the Euler totient function.) It is well known that if the prime decomposition of $n$

is given by $n = p_1^{e_1} p_2^{e_2} \cdots p_k^{e_k}$, then

$$\varphi(n) = p_1^{e_1-1}(p_1 - 1)p_2^{e_2-1}(p_2 - 1) \cdots p_k^{e_k-1}(p_k - 1).$$

## 2.1 Probabilistic Encryption

In 1984, Goldwasser and Micali ([GoMi84]) introduced the notion of *probabilistic encryption.* A probabilistic encryption method allows one to encrypt a fixed value in many different ways. Thus, even when given the encryption of a value and details of the encryption mechanism, it is not necessarily possible for an adversary to determine whether or not the encryption represents the encryption of a chosen value.

Goldwasser and Micali develop a bit encryption function based on the problem of quadratic residuosity.

**Definition** An integer $y$ is a *quadratic residue* modulo an integer $n$ if and only if there exists an integer $x$ such that $y \equiv_n x^2$. Let $\mathbf{Z_n^2}$ denote the set of quadratic residues modulo $n$ which are relatively prime to $n$.

Let $n$ be an integer which is the product of two distinct odd primes $p$ and $q$. An integer $z$ which is relatively prime to $n$ is said to be of Jacobi symbol $+1$ if $z \in \mathbf{Z_p^2}$ and $z \in \mathbf{Z_q^2}$ or if $z \notin \mathbf{Z_p^2}$ and $z \notin \mathbf{Z_q^2}$. Otherwise, $z$ is said to be of Jacobi symbol $-1$. Every quadratic residue in $\mathbf{Z_n^*}$ clearly has Jacobi symbol $+1$.

There is a simple effective polynomial time procedure originally due to Gauss ([Gaus01]) which computes the Jacobi symbol of an integer with respect to a given modulus. There is, however, no known polynomial time procedure to, without the factorization of $n$, determine whether or not an integer with Jacobi symbol $+1$ is a quadratic residue modulo $n$. In addition, given a single integer $y$ of Jacobi symbol $+1$ which is not in $\mathbf{Z_n^2}$, it is possible to uniformly select quadratic residues or quadratic non-residues modulo $n$, even if the factorization of $n$ is not known.

Thus, a probabilistic public-key bit encryption function proposed by Goldwasser and Micali can be defined by a user by selecting an $n$ of known factorization $n = pq$ where $p$ and $q$ are distinct odd primes and releasing this $n$ together with a $y$ of Jacobi symbol $+1$ which is not in $\mathbf{Z_n^2}$. An encrypted bit may be sent to this user by releasing a quadratic residue to indicate a zero or a quadratic

non-residue of Jacobi symbol +1 to indicate a one. The user which possesses the factorization of $n$ can easily determine which is the case (see [NiZu72], for instance). Without the factorization, however, distinguishing between the two cases is an apparently difficult problem of unknown complexity.

Goldwasser and Micali show that any non-trivial information that an adversary can effectively derive from even the repeated use of such a function would yield an effective procedure for distinguishing residues from non-residues, and no such procedure is known.

## 2.2 Higher Residues

For the purposes of this work, it is useful to generalize the Goldwasser and Micali encryption function to a function which can encrypt more than one bit at a time. To accomplish this, higher residues are used.

**Definition** For a given integer $n$, an integer $z$ is said to be an $r^{\text{th}}$ *residue* modulo $n$ if and only if there exists some integer $x$ such that $z \equiv_n x^r$. Let $\mathbf{Z_n^r}$ denote the set of $r^{\text{th}}$ residues modulo $n$ which are relatively prime to $n$, and denote by $\overline{\mathbf{Z_n^r}}$ the set of $z \in \mathbf{Z_n^*}$ which are *not* $r^{\text{th}}$ residues modulo $n$.

**Lemma 2.1** $\mathbf{Z_n^r}$ *is a subgroup of* $\mathbf{Z_n^*}$.

*Proof:*

To see that $\mathbf{Z_n^r}$ is a closed, consider $z_1, z_2 \in \mathbf{Z_n^r}$ with $z_1 \equiv_n x_1^r$ and $z_2 \equiv_n x_2^r$. The product $z_1 z_2 \equiv_n x_1^r x_2^r \equiv_n (x_1 x_2)^r$ is an $r^{\text{th}}$ residue modulo $n$ and is relatively prime to $n$ since both $z_1$ and $z_2$ are by definition relatively prime to $n$.

To see that inverses exist, simply note that if $z \equiv_n x^r$, then $(x^{-1})^r \in \mathbf{Z_n^r}$ is the inverse of $z$.

Of course, $1^r \equiv_n 1$ so $1 \in \mathbf{Z_n^r}$ for all $r$.

Finally, associativity is inherited from the group $\mathbf{Z_n^*}$ which in turn inherits its associativity from integer multiplication. ∎

**Lemma 2.2** *Given a fixed* $r$ *and* $n$, *every integer* $z \in \mathbf{Z_n^r}$ *has the same number of* $r^{\text{th}}$ *roots.*

*Proof:*

Let $\zeta_1, \zeta_2, \ldots, \zeta_k$ be the distinct $r^{\text{th}}$ roots of 1. Since $1^r \equiv_n 1$, $1 \in \mathbf{Z}_n^r$ and $k \geq 1$. Next consider a $z \in \mathbf{Z}_n^r$. By definition, there exists some $x \in \mathbf{Z}_n^*$ such that $z \equiv_n x^r$. Since $\mathbf{Z}_n^*$ is a group, $x\zeta_i$ are distinct for distinct $i$. Also, $(x\zeta_i)^r \equiv_n x^r\zeta_i^r \equiv_n z$. Thus, the $x\zeta_i$ form $k$ distinct $r^{\text{th}}$ roots of $z$. If $z$ had some additional $r^{\text{th}}$ root $\overline{x}$ not among the $x\zeta_i$, then $(\overline{x}x^{-1})^r \equiv_n \overline{x}^r x^{-r} \equiv_n zz^{-1} \equiv_n 1$. Thus, $\overline{x}x^{-1}$ is an $r^{\text{th}}$ root of 1 and must be congruent to some $\zeta_i$. But if $\overline{x}x^{-1} \equiv_n \zeta_i$, then $\overline{x} \equiv_n x\zeta_i$, and this violates the assumption on $\overline{x}$. Thus, every $z \in \mathbf{Z}_n^r$ has the same number of $r^{\text{th}}$ roots as does 1. ∎

**Lemma 2.3** *If $r$ and $\varphi(n)$ are relatively prime, then every integer $z \in \mathbf{Z}_n^*$ is an $r^{\text{th}}$ residue modulo $n$ (i.e. $\mathbf{Z}_n^r = \mathbf{Z}_n^*$), and an $r^{\text{th}}$ root of $z$ is given by $z^A \bmod n$ where $A$ satisfies $Ar - B\varphi(n) = 1$.*

*Proof:*

Since $r$ and $\varphi(n)$ are relatively prime, the Diophantine equation $Ar - B\varphi(n) = 1$ has integral solutions $A$ and $B$ which can easily be computed by the extended Euclidean algorithm when the factorization of $n$ (and hence $\varphi(n)$) is known.

Since $z$ is relatively prime to $n$, $z^{\varphi(n)} \equiv_n 1$ (by Euler's theorem). Therefore,

$$(z^A)^r \equiv_n z^{B\varphi(n)+1} \equiv_n (z^{\varphi(n)})^B z \equiv_n z.$$

Thus, $z^A$ is a $r^{\text{th}}$ root of $z$ modulo $n$. ∎

Lemma 2.3 serves as the basis for the RSA public-key cryptosystem (see [RSA78]).

## 2.3 Residue Classes

**Definition** Let $r$, $n$, and $y$ be fixed integers. If $w \in \mathbf{Z}_n^*$ is expressible as $w \equiv_n y^c z$ for some $z \in \mathbf{Z}_n^r$, then $w$ is said to be of *residue class* $c$ with respect to $r$, $n$, and $y$. The set of all elements of $\mathbf{Z}_n^*$ which are of residue class $c$ (again with respect to a fixed $r$, $n$, and $y$) is denoted by $\mathcal{RC}[c]_{(r,n,y)}$ (or simply $\mathcal{RC}[c]$ when $r$, $n$, and $y$ are understood). In particular, it is clear that for all $y \in \mathbf{Z}_n^*$, $\mathcal{RC}[0]_{(r,n,y)} = \mathbf{Z}_n^r$.

**Lemma 2.4** *Let $(r,n,y)$ be integers with $y \in \mathbf{Z}_n^*$. If $\mathcal{RC}[c_1] \cap \mathcal{RC}[c_2]$ is not empty, then $\mathcal{RC}[c_1] = \mathcal{RC}[c_2]$.*

*Proof:*

Let $w \in \mathcal{RC}[c_1] \cap \mathcal{RC}[c_2]$. By definition, $w \equiv_n y^{c_1} z_1 \equiv_n y^{c_2} z_2$ for $z_1, z_2 \in \mathbf{Z}_n^r$. Since $\mathbf{Z}_n^r$ is a group, this implies that $y^{c_1 - c_2} \equiv_n z_2 z_1^{-1} \in \mathbf{Z}_n^r$.

Let $w' \in \mathcal{RC}[c_1]$. By definition, $w' \equiv_n y^{c_1} z_1'$ for some $z_1' \in \mathbf{Z}_n^r$. Let $z_2' \equiv_n z_1' y^{c_1 - c_2} \in \mathbf{Z}_n^r$. Thus, $z_1' \equiv_n y^{c_2 - c_1} z_2'$. Now,

$$w' \equiv_n y^{c_1} z_1' \equiv_n y^{c_1} y^{c_2 - c_1} z_2' \equiv_n y^{c_2} z_2'.$$

Since $z_2' \in \mathbf{Z}_n^r$, $w' \in \mathcal{RC}[c_2]$. Thus, $\mathcal{RC}[c_1] \subseteq \mathcal{RC}[c_2]$. By symmetry, therefore, $\mathcal{RC}[c_1] = \mathcal{RC}[c_2]$, as desired. ∎

Lemma 2.4 implies that for a given $r$, $n$, and $y$, any two residue classes either coincide or are disjoint.

**Definition** Given a triple of integers $(r,n,y)$, the *norm* of $(r,n,y)$ (written $|(r,n,y)|$) is the least positive integer $m$ such that $y^m \in \mathbf{Z}_n^r$. If no such integer exists, then $|(r,n,y)| = \infty$.

When $y \notin \mathbf{Z}_n^*$, then $|(r,n,y)| = \infty$ since $\mathbf{Z}_n^r$ consists of only those $r^{\text{th}}$ residues which are relatively prime to $n$. Whenever $y \in \mathbf{Z}_n^*$, however, $|(r,n,y)|$ is bounded by $r$ since $y^r \in \mathbf{Z}_n^r$.

**Lemma 2.5** *Let $(r,n,y)$ be integers with $y \in \mathbf{Z}_n^*$. The norm $m = |(r,n,y)|$ is a divisor of $r$.*

*Proof:*

Since $y^r \in \mathbf{Z}_n^r$, $m \leq r$. By definition, $y^m \in \mathbf{Z}_n^r$. Let $g = \gcd(m,r)$. There exist integral solutions to the diophantine equation $Am + Br = g$. Thus,

$$y^g \equiv_n y^{Am+Br} \equiv_n (y^m)^A (y^r)^B \in \mathbf{Z}_n^r.$$

Since $g$ is a positive integer and $g \leq m$, it must be the case that $g = m$ (since $m$ is the *least* positive integer such that $y^m \in \mathbf{Z}_n^r$). Thus, since $g \mid r$, it is true that $m \mid r$, as desired. ∎

The next lemma describes the number of distinct residue classes.

**Lemma 2.6** *Let $(r,n,y)$ be integers with $y \in \mathbf{Z}_n^*$, and let $m = |(r,n,y)|$. $\mathcal{RC}[c_1] = \mathcal{RC}[c_2]$ if and only if $c_1 \equiv_m c_2$.*

*Proof:* By definition, $y^{c_1} \in \mathcal{RC}[c_1]$ and $y^{c_2} \in \mathcal{RC}[c_2]$. If $c_1 \equiv_m c_2$, then there exists some integer $a$ such that $am = c_1 - c_2$. Since $m = |(r, n, y)|$ and $y \in \mathbf{Z}_n^*$, $y^m \in \mathbf{Z}_n^r$, and therefore, $y^{am} \in \mathbf{Z}_n^r$. Thus,

$$y^{c_1} \equiv_n y^{c_2} y^{c_1 - c_2} \equiv_n y^{c_2} y^{am} \in \mathcal{RC}[c_2]$$

since $y^{am} \in \mathbf{Z}_n^r$. Hence $y^{c_1} \in \mathcal{RC}[c_1] \cap \mathcal{RC}[c_2]$. So by lemma 2.4, $\mathcal{RC}[c_1] = \mathcal{RC}[c_2]$.

Conversely, if $\mathcal{RC}[c_1] = \mathcal{RC}[c_2]$, then since $y^{c_1} \in \mathcal{RC}[c_1]$, $y^{c_1} \in \mathcal{RC}[c_2]$. Thus, there exists a $z \in \mathbf{Z}_n^r$ such that $y^{c_1} \equiv_n y^{c_2} z$. Therefore, $y^{c_1 - c_2} \in \mathbf{Z}_n^r$. Let $g = \gcd(m, c_1 - c_2)$. There exist integers $A$ and $B$ such that $g = Am + B(c_1 - c_2)$. Now,

$$y^g \equiv_n y^{Am + B(c_1 - c_2)} \equiv_n (y^m)^A (y^{c_1 - c_2})^B \in \mathbf{Z}_n^r$$

since $y^m \in \mathbf{Z}_n^r$ by definition and $y^{c_1 - c_2} \in \mathbf{Z}_n^r$ by the above argument. Thus, since $g \mid m$ and $y^g \in \mathbf{Z}_n^r$, it must be the case that $g = m$. But it is also the case that $g$ (and hence $m$) divides $c_1 - c_2$. Therefore, $c_1 \equiv_m c_2$, as desired. ∎

Thus, given a triple $(r, n, y)$ with $y \in \mathbf{Z}_n^*$, there are exactly $m = |(r, n, y)|$ distinct residue classes and they are given by

$$\mathbf{Z}_n^r = \mathcal{RC}[0], \mathcal{RC}[1], \ldots, \mathcal{RC}[m-1].$$

This justifies the terminology of a "residue class".

The following lemma gives a method for selecting uniformly from elements of a given residue class $c$.

**Lemma 2.7** *Let $(r, n, y)$ be integers with $y \in \mathbf{Z}_n^*$. If $x$ is chosen uniformly from $\mathbf{Z}_n^*$, then $w = y^c x^r \bmod n$ represents a uniformly chosen member of $\mathcal{RC}[c]$.*

*Proof:*

Since $\mathbf{Z}_n^*$ is a group and $c$ is fixed, $y^c$ is fixed and $w \equiv_n y^c z_1 \equiv_n y^c z_2$ if and only if $z_1 \equiv_n z_2$. Therefore, each $w \in \mathcal{RC}[c]$ is expressible as $w \equiv_n y^c z$ for *exactly* one $z \in \mathbf{Z}_n^r$. (Recall that, by definition, there must be at least one such $z$.) By lemma 2.2, every $z \in \mathbf{Z}_n^r$ has exactly has the same number of distinct $r^{\text{th}}$ roots in $\mathbf{Z}_n^*$. Thus, if $x$ is chosen uniformly from $\mathbf{Z}_n^*$, $x^r$ is a uniformly chosen element of $\mathbf{Z}_n^r$, and $y^c x^r$ is a uniformly chosen element of $\mathcal{RC}[c]$. ∎

The essence of Lemma 2.7 is captured in more general terms by Angluin and Lichtenstein in [AnLi83] as the so called property of *random self-reducibility.*

They point out that this property is common to a wide variety of number-theoretic problems and "can be used to show a problem is uniformly hard if it is hard at all".

The following lemmas give some important properties of residue classes.

**Lemma 2.8** *Let $(r, n, y)$ be integers with $y \in \mathbf{Z}_n^*$. If $w_1 \in \mathcal{RC}[c_1]$ and $w_2 \in \mathcal{RC}[c_2]$, then the product $w_1 w_2 \in \mathcal{RC}[c_1 + c_2]$.*

*Proof:*

Each $w_i \in \mathcal{RC}[c_i]$ if and only if it is expressible as $w_i \equiv_n y^{c_i} z_i$ for some $z_i \in \mathbf{Z}_n^r$. Thus, $w_1 w_2 \equiv_n (y^{c_1} z_1)(y^{c_2} z_2) \equiv_n y^{c_1 + c_2} z$ where $z \equiv_n z_1 z_2 \in \mathbf{Z}_n^r$. Thus, $w_1 w_2 \in \mathcal{RC}[c_1 + c_2]$. ∎

**Lemma 2.9** *Let $(r, n, y)$ be integers with $y \in \mathbf{Z}_n^*$. If $w \in \mathcal{RC}[c]$, then $w^{-1} \in \mathcal{RC}[-c]$.*

*Proof:*

$w \in \mathcal{RC}[c]$ if and only if it is expressible as $w \equiv_n y^c z$ for some $z \in \mathbf{Z}_n^r$. Since $w \in \mathbf{Z}_n^*$, $w^{-1}$ is well-defined and $w^{-1} \equiv_n y^{-c} z^{-1}$. But since $\mathbf{Z}_n^r$ is a group, $z^{-1} \in \mathbf{Z}_n^r$. Thus, by definition, $w^{-1} \in \mathcal{RC}[-c]$. ∎

The following lemma shows how two integers can be shown to be of the same residue class.

**Lemma 2.10** *Let $(r, n, y)$ be integers with $y \in \mathbf{Z}_n^*$. Two integers $w_1$ and $w_2$ are of the same residue class with respect to $r$, $n$, and $y$ if and only if $w_1 w_2^{-1} \in \mathbf{Z}_n^r$.*

*Proof:*

If $w_1$ and $w_2$ are both in $\mathcal{RC}[c]$, then by lemma 2.9, $w_2^{-1}$ is in $\mathcal{RC}[-c]$ and by lemma 2.8, $w_1 w_2^{-1}$ is in $\mathcal{RC}[0]$. But this implies that $w_1 w_2^{-1} \in \mathbf{Z}_n^r$, as desired.

Conversely, assume that $w \equiv_n w_1 w_2^{-1} \in \mathbf{Z}_n^r = \mathcal{RC}[0]$. Since $w \in \mathbf{Z}_n^r$, so is $w^{-1}$ and so $w^{-1} \in \mathcal{RC}[0]$. If $w_2 \in \mathcal{RC}[c]$, then $w_1 \equiv_n w_2 w$ is also in $\mathcal{RC}[c]$ by lemma 2.8. Similarly, if $w_1 \in \mathcal{RC}[c]$, then $w_2 \equiv_n w_1 w^{-1} \in \mathcal{RC}[c]$. ∎

Thus, to prove that two integers are of the same residue class, it is necessary only to exhibit an $r^{\text{th}}$ root of their quotient.

## 2.4 The Consonance Property

**Definition** A triple of positive integers $(r, n, y)$ is said to be *consonant* (or simply $r$, $n$, and $y$ are *consonant*) if $|(r, n, y)| = r$.

The most important feature of a consonant triple $(r, n, y)$ is that the residue classes $\mathcal{RC}[0], \mathcal{RC}[1], \ldots, \mathcal{RC}[r-1]$ are all distinct.

**Lemma 2.11** *Let $y \in \mathbf{Z}_n^*$. There exist $r$ distinct residue classes if and only if $(r, n, y)$ is consonant. In addition, if $(r, n, y)$ is not consonant, then the number of distinct residue classes is at most $r/2$.*

*Proof:*

By lemma 2.6 and the definition of consonant, when $(r, n, y)$ is consonant, there exist $r$ distinct residue classes.

Conversely, since $y \in \mathbf{Z}_n^*$, $y^r \in \mathbf{Z}_n^r$, and the norm and hence the number of distinct residue classes is bounded by $r$. If the norm were exactly $r$, then $(r, n, y)$ would be consonant. If $(r, n, y)$ is not consonant, then by lemma 2.5, the norm must be a proper divisor of $r$. This implies that the norm of $(r, n, y)$ (and hence the number of distinct residue classes) is bounded by $r/2$. ∎

### 2.4.1 Characterizations of Consonance

We now begin to characterize what it means for a triple to be consonant.

**Lemma 2.12** *A triple $(r, n, y)$ is consonant if and only if the following condition holds for all integers $c$:*

$$y^c \in \mathbf{Z}_n^r \iff c \equiv_r 0.$$

*Proof:*

If $(r, n, y)$ is consonant, then by lemma 2.6,

$$c \equiv_r 0 \Rightarrow \mathcal{RC}[c] = \mathcal{RC}[0] = \mathbf{Z}_n^r.$$

Thus, in particular, $y^c \in \mathbf{Z}_n^r$.

Also, if $(r, n, y)$ is consonant, then

$$y^c \in \mathbf{Z}_n^r \Rightarrow y^c \in \mathcal{RC}[c] \cap \mathcal{RC}[0] \Rightarrow \mathcal{RC}[c] = \mathcal{RC}[0] \Rightarrow c \equiv_r 0$$

by lemmas 2.4 and 2.6.

Conversely, assume that

$$y^c \in \mathbf{Z}_\mathbf{n}^\mathbf{r} \iff c \equiv_r 0.$$

Since this condition must hold for all $c$, consider the case of $c = r$. Here, $y^r \in \mathbf{Z}_\mathbf{n}^\mathbf{r}$. But $\mathbf{Z}_\mathbf{n}^\mathbf{r}$ is defined to be a subset of $\mathbf{Z}_\mathbf{n}^*$. Thus, $y^r \in \mathbf{Z}_\mathbf{n}^*$, and therefore, $y \in \mathbf{Z}_\mathbf{n}^*$. This implies that the norm $m = |(r, n, y)|$ is finite. Next, by lemma 2.6, it is true that for all integers $c$

$$c \equiv_m 0 \Rightarrow \mathcal{RC}[c] = \mathcal{RC}[0] \Rightarrow y^c \in \mathbf{Z}_\mathbf{n}^\mathbf{r}.$$

But the premise gives that

$$y^c \in \mathbf{Z}_\mathbf{n}^\mathbf{r} \Rightarrow c \equiv_r 0.$$

Thus,

$$c \equiv_m 0 \Rightarrow c \equiv_r 0.$$

Hence $r \mid m$ must be true. But by lemma 2.5, $m \mid r$ must also be true. Therefore, $m = r$, as desired. ∎

We now give necessary conditions for a triple $(r, n, y)$ to be consonant.

**Theorem 2.13** *If $(r, n, y)$ is consonant, then the following are true.*

*(a)* $y \in \mathbf{Z}_\mathbf{n}^*$.

*(b) If $q$ is a prime such that $q \mid r$, then* $y \notin \mathbf{Z}_\mathbf{n}^\mathbf{q}$.

*(c) If $q$ is a prime such that $q^\alpha \mid r$, then there exists a prime power $p^e \mid n$ such that $q^\alpha \mid \varphi(p^e)$ and* $y \notin \mathbf{Z}_\mathbf{p^e}^\mathbf{q}$.

*Proof:*

(a) When $y \notin \mathbf{Z}_\mathbf{n}^*$, $|(r, n, y)| = \infty$. Thus, $|(r, n, y)| = r$ implies that $y \in \mathbf{Z}_\mathbf{n}^*$.

(b) If $q$ is a prime such that $q \mid r$, then if $y \in \mathbf{Z}_\mathbf{n}^\mathbf{q}$ were true, there would exist some $x \in \mathbf{Z}_\mathbf{n}^*$ such that $y \equiv_n x^q$. This would imply that $y^{r/q} \equiv_n (x^q)^{r/q} \equiv_n x^r$, and therefore that $y^{r/q} \in \mathbf{Z}_\mathbf{n}^\mathbf{r}$. But this would violate lemma 2.12. Hence, $y \notin \mathbf{Z}_\mathbf{n}^\mathbf{q}$.

(c) Fix $q^\alpha$ to be a prime power such that $q^\alpha \mid r$ and consider $y^{r/q}$. If for each prime power $p^e \mid n$, it were true that $y^{r/q} \in \mathbf{Z}_\mathbf{p^e}^\mathbf{r}$, then (by the definition of $\mathbf{Z}_\mathbf{p^e}^\mathbf{r}$) $y^{r/q}$ would have an $r^{\text{th}}$ root modulo each $p^e$. The Chinese Remainder Theorem

would then imply that $y^{r/q}$ would have an $r^{\text{th}}$ root modulo $n$, and therefore that $y^{r/q} \in \mathbf{Z}_{\mathbf{n}}^{\mathbf{r}}$. But this contradicts lemma 2.12, and hence there must be at least one prime power $p^e \mid n$ such that $y^{r/q} \notin \mathbf{Z}_{\mathbf{p^e}}^{\mathbf{r}}$.

Now, fix $p^e$ to be a prime power divisor of $n$ such that $y^{r/q} \notin \mathbf{Z}_{\mathbf{p^e}}^{\mathbf{r}}$, and let $g = \gcd(r, \varphi(p^e))$. Let $A$ and $B$ be integers such that $g = Ar + B\varphi(p^e)$. Then

$$y^g \equiv_{p^e} y^{Ar+B\varphi(p^e)} \equiv_{p^e} (y^A)^r \in \mathbf{Z}_{\mathbf{p^e}}^{\mathbf{r}}.$$

If $q^\alpha \nmid \varphi(p^e)$ were true, then (since $q^\alpha \mid r$) $g = \gcd(r, \varphi(p^e)) = \gcd(r/q, \varphi(p^e))$ and hence $g$ would be a divisor of $r/q$. But this would imply that $y^{r/q} \in \mathbf{Z}_{\mathbf{p^e}}^{\mathbf{r}}$. Hence, $q^\alpha \mid \varphi(p^e)$ and $y^{r/q} \notin \mathbf{Z}_{\mathbf{p^e}}^{\mathbf{r}}$.

Finally, if $y \in \mathbf{Z}_{\mathbf{p^e}}^{\mathbf{q}}$ were so, then (since a $q^{\text{th}}$ root of $y$ is an $r^{\text{th}}$ root of $y^{r/q}$) $y^{r/q} \in \mathbf{Z}_{\mathbf{p^e}}^{\mathbf{r}}$ would also be true. But this cannot be the case. Hence, $y \notin \mathbf{Z}_{\mathbf{p^e}}^{\mathbf{q}}$. ∎

Two important corollaries to theorem 2.13 will be given. The first follows easily from theorem 2.13(c) and the second from theorem 2.13(b). However in both cases, direct proofs seem to be even simpler.

**Corollary 2.14** *If $(r, n, y)$ is consonant, then $r \mid \varphi(n)$.*

*Proof:*

Let $g = \gcd(r, \varphi(n))$ There exist integers $A$ and $B$ such that $g = Ar + B\varphi(n)$.

$$y^g \equiv_n y^{Ar+B\varphi(n)} \equiv_n (y^A)^r \in \mathbf{Z}_{\mathbf{n}}^{\mathbf{r}}.$$

Thus, since $(r, n, y)$ is consonant, $g \equiv_r 0$. This implies that $g = r$ and hence that $r \mid \varphi(n)$. ∎

**Corollary 2.15** *If $(r, n, y)$ is consonant and $r > 1$, then $y \notin \mathbf{Z}_{\mathbf{n}}^{\mathbf{r}}$.*

*Proof:*

$y \in \mathbf{Z}_{\mathbf{n}}^{\mathbf{r}} \Rightarrow 1 \equiv_r 0 \Rightarrow r = 1.$ ∎

We now show that conditions (a)–(c) of theorem 2.13 together with one added condition are sufficient for a triple $(r, n, y)$ to be consonant.

**Theorem 2.16** *A triple of positive integers $(r, n, y)$ is consonant if all of the following are true.*

*(a)* $y \in \mathbf{Z}_{\mathbf{n}}^{*}$.

*(b) If $q$ is a prime such that $q \mid r$, then* $y \notin \mathbf{Z}_{\mathbf{n}}^{\mathbf{q}}$.

*(c) If $q$ is a prime such that $q^\alpha \mid r$, then there exists a prime power $p^e \mid n$ such that $q^\alpha \mid \varphi(p^e)$ and $y \notin \mathbf{Z}^{\mathbf{q}}_{\mathbf{p^e}}$.*

*(d) $r$ and $n$ are not both even.*

*Proof:*

By lemma 2.12, a triple $(r, n, y)$ is consonant if and only if for all integers $c$,

$$y^c \in \mathbf{Z}^{\mathbf{r}}_{\mathbf{n}} \iff c \equiv_r 0.$$

Thus, to show that $(r, n, y)$ is consonant, it is sufficient to show that

$$c \equiv_r 0 \Rightarrow y^c \in \mathbf{Z}^{\mathbf{r}}_{\mathbf{n}}$$

and

$$y^c \in \mathbf{Z}^{\mathbf{r}}_{\mathbf{n}} \Rightarrow c \equiv_r 0.$$

If $c \equiv_r 0$, then $c = ar$ for some integer $a$, and therefore $y^c \equiv_n (y^a)^r \in \mathbf{Z}^{\mathbf{r}}_{\mathbf{n}}$.

We must now show only that conditions (a)–(d) together with the assumption that $y^c \in \mathbf{Z}^{\mathbf{r}}_{\mathbf{n}}$ imply that $c \equiv_r 0$. Suppose that there exists a triple $(r, n, y)$ such that conditions (a)–(d) are true, that $y^c \in \mathbf{Z}^{\mathbf{r}}_{\mathbf{n}}$, but that $c \not\equiv_r 0$. Since $r \nmid c$, there exists some prime $q$ and a positive integer $\alpha$ such that $q^\alpha \mid r$, but $q^\alpha \nmid c$. By condition (c), there exists a prime power $p^e \mid n$ such that $q^\alpha \mid \varphi(p^e)$ and $y \notin \mathbf{Z}^{\mathbf{q}}_{\mathbf{p^e}}$.

If $n$ is odd, then since $p^e \mid n$, $p$ is odd. If $n$ is even, then by condition (d), $r$ is odd and therefore since $q^\alpha \mid r$, $q$ is odd. But the fact that $q$ is an odd prime which divides $\varphi(p^e)$ implies that $p$ is odd (since $\varphi(2^e) = 2^{e-1}$). Thus, whether $n$ is even or odd, we may assume that $p$ is an odd prime and therefore that $\mathbf{Z}^*_{\mathbf{p^e}}$ is cyclic. Let $g$ be a generator of this group.

Write $y$ as $y \equiv_{p^e} g^a$, for some integer $a$. $y^c \in \mathbf{Z}^{\mathbf{r}}_{\mathbf{n}}$ by assumption, and therefore $y^c \in \mathbf{Z}^{\mathbf{r}}_{\mathbf{p^e}}$. Therefore, $g^{ac} \equiv_{p^e} y^c \in \mathbf{Z}^{\mathbf{q}^\alpha}_{\mathbf{p^e}}$. Thus, since $q^\alpha \mid \varphi(p^e)$, $q^\alpha \mid ac$. But since $q^\alpha \nmid c$ and $q$ is prime, $q \mid a$. Therefore, $y \equiv_{p^e} g^a \in \mathbf{Z}^{\mathbf{q}}_{\mathbf{p^e}}$. But we have already shown (under the assumption that $c \not\equiv_r 0$) that $y \notin \mathbf{Z}^{\mathbf{q}}_{\mathbf{p^e}}$. This is a contradiction, and hence, $c \equiv_r 0$ as desired. ∎

We have not completely characterized the cases when $(r, n, y)$ is consonant. When $r$ and $n$ are both even, we have not seen the precise properties necessary and sufficient for $(r, n, y)$ to be consonant. Although this case is not of great interest, the results are stated for completeness. The proofs are somewhat complex and are omitted.

**Claim** If $r$ is even but $4 \nmid r$, then $(r, n, y)$ is consonant exactly when the conditions (a)–(c) of theorems 2.13 and 2.16 are met.
Similarly, if $n$ is even but $8 \nmid n$, then $(r, n, y)$ is consonant exactly when the conditions (a)–(c) of theorems 2.13 and 2.16 are met.
If $4 \mid r$ and $8 \mid n$, then let $\alpha$ be the greatest integer such that $2^\alpha \mid r$. In this case, $(r, n, y)$ is consonant exactly when $2^{\alpha+2} \mid n$, the conditions (a)–(c) of theorem 2.13 and 2.16 are met, and $y \equiv_8 \pm 3$.

In particular, $(2, n, y)$ is consonant whenever $y \in \mathbf{Z}_\mathbf{n}^*$ and $y \notin \mathbf{Z}_\mathbf{n}^\mathbf{2}$. This is the case used in the Goldwasser–Micali probabilistic encryption method ([GoMi84]) described in section 2.1.

**Theorem 2.17** *If $(r, n, y)$ is consonant, then each $w \in \mathbf{Z}_\mathbf{n}^*$ is expressible as $w \equiv_n y^c z$ for at most one integer $c$ such that $0 \leq c < r$ and at most one $z \in \mathbf{Z}_\mathbf{n}^\mathbf{r}$.*

*Proof:*

Suppose $w \equiv_n y^{c_1} z_1 \equiv_n y^{c_2} z_2$ for $0 \leq c_1, c_2 < r$ and $z_1, z_2 \in \mathbf{Z}_\mathbf{n}^\mathbf{r}$. Since $y, z_1, z_2 \in \mathbf{Z}_\mathbf{n}^*$, we can form $y^{c_1 - c_2} \equiv_n z_2 z_1^{-1}$, and since $z_1, z_2 \in \mathbf{Z}_\mathbf{n}^\mathbf{r}$ and $\mathbf{Z}_\mathbf{n}^\mathbf{r}$ is a group, $y^{c_1 - c_2} \equiv_n z_2 z_1^{-1} \in \mathbf{Z}_\mathbf{n}^\mathbf{r}$. Since $0 \leq c_1, c_2 < r$, $|c_1 - c_2| < r$. Since $(r, n, y)$ is consonant, $r \mid (c_1 - c_2)$ and hence $c_1 = c_2$. Finally, since $\mathbf{Z}_\mathbf{n}^*$ is a group, $c_1 = c_2$ implies that $z_1 \equiv_n z_2$. ∎

Thus, when $(r, n, y)$ is consonant the residue classes

$$\mathcal{RC}[0], \mathcal{RC}[1], \ldots, \mathcal{RC}[r-1]$$

are all distinct and, by lemma 2.11, represent the entire set of residue classes.

### 2.4.2 Prime Consonance

A particularly useful special case of consonance occurs when the first component $r$ of a consonant triple $(r, n, y)$ is prime.

**Definition** A triple $(r, n, y)$ is said to be *prime consonant* if $r$ is prime and if $(r, n, y)$ is consonant.

**Theorem 2.18** *A triple $(r, n, y)$ is prime consonant if and only if $r$ is a prime and $y \in \overline{\mathbf{Z}_\mathbf{n}^\mathbf{r}}$.*

*Proof:*

If $(r, n, y)$ is prime consonant, then $r$ is prime (by definition). Also, $y \in \mathbf{Z}_n^*$ (by theorem 2.13(a)), and $y \notin \mathbf{Z}_n^r$ (by theorem 2.13(b)). Thus, $y \in \overline{\mathbf{Z}_n^r}$.

Conversely, if $r$ is prime then either $r \mid \varphi(n)$ or $r$ and $\varphi(n)$ are relatively prime. If $r$ and $\varphi(n)$ were relatively prime, then by lemma 2.3 $\mathbf{Z}_n^* = \mathbf{Z}_n^r$ would be true. But the assumption that $y \in \overline{\mathbf{Z}_n^r}$ implies that $y \in \mathbf{Z}_n^*$ and $y \notin \mathbf{Z}_n^r$. Thus, $r \mid \varphi(n)$. If $r$ is odd the conditions of theorem 2.16 are easily satisfied. If $r = 2$, then since $y \notin \mathbf{Z}_n^r$ and $y^2 \in \mathbf{Z}_n^r$, $|(r, n, y)| = 2$, and therefore $(r, n, y)$ is consonant. ∎

### 2.4.3 Perfect Consonance

The next lemma shows how $r^{\text{th}}$ roots modulo $n$ can be computed when $r \mid \varphi(n)$, $r$ and $\varphi(n)/r$ are relatively prime, and the factorization of $n$ is known.

**Lemma 2.19** *If $r \mid \varphi(n)$, $r$ and $\varphi(n)/r$ are relatively prime, and $z \in \mathbf{Z}_n^r$, then an $r^{\text{th}}$ root of $z$ modulo $n$ is given by $z^A \bmod n$ where $A$ satisfies $Ar - B\varphi(n)/r = 1$.*

*Proof:*

Since $r \mid \varphi(n)$ and since $r$ and $\varphi(n)/r$ are relatively prime, the Diophantine equation $Ar - B\varphi(n)/r = 1$ has integral solutions $A$ and $B$, and such solutions can easily be computed by the extended Euclidean algorithm when the factorization of $n$ (and hence $\varphi(n)$) is known.

Since $z \in \mathbf{Z}_n^r$, $z \equiv_n x^r$ for some $x \in \mathbf{Z}_n^*$. Therefore,

$$(z^A)^r \equiv_n (x^{Ar})^r \equiv_n (x^{B\varphi(n)/r+1})^r \equiv_n x^{B\varphi(n)+r} \equiv_n x^r \equiv_n z$$

since $a^{\varphi(n)} \equiv_n 1$ for all $a \in \mathbf{Z}_n^*$.

Thus $z^A$ is an $r^{\text{th}}$ root of $z$. ∎

**Theorem 2.20** *If $r \mid \varphi(n)$, $r$ and $\varphi(n)/r$ are relatively prime, and $z \in \mathbf{Z}_n^r$, then $z$ has* exactly *$r$ distinct $r^{\text{th}}$ roots.*

*Proof:*

Let $n = p_1^{e_1} p_2^{e_2} \cdots p_k^{e_k}$ be the prime factorization of $n$. The multiplicative property of the Euler totient function gives $\varphi(n) = \varphi(p_1^{e_1})\varphi(p_2^{e_2}) \cdots \varphi(p_k^{e_k})$. Let $h_i = \gcd(\varphi(p_i^{e_i}), r)$. We will first show that modulo each $p_i^{e_i}$ there are *exactly* $h_i$ $r^{\text{th}}$ roots of 1.

When $p$ is an odd prime, $\mathbf{Z}^*_{\mathbf{p^e}}$ is cyclic. Let $g$ be a generator of this group. Thus, the $\varphi(p^e)$ elements of $\mathbf{Z}^*_{\mathbf{p^e}}$ are expressible as $g^a$ as $a$ ranges in $0 \leq a < \varphi(p^e)$. Now, an element $g^a$ is an $r^{\text{th}}$ root of 1 if and only if $g^{ar} \equiv_{p^e} 1$ which is the case if and only if $\varphi(p^e) \mid ar$. But this is the case precisely when $a$ is a multiple of $\varphi(p^e)/\gcd(\varphi(p^e), r)$. Thus, the number of integers $a$ in the range $0 \leq a < \varphi(p^e)$ for which $g^a$ is an $r^{\text{th}}$ root of 1 is exactly $\gcd(\varphi(p^e), r)$. Hence, when $p_i$ is odd, the number of $r^{\text{th}}$ roots of 1 modulo $p_i^{e_i}$ is exactly $h_i$.

When $p = 2$ and $2^e$ is the largest power of 2 which divides $n$, then there are two cases. If $r$ is even, then the fact that $r$ and $\varphi(n)/r$ are relatively prime implies that $\varphi(n)/r$ is odd. Since $\varphi$ is multiplicative and $\varphi(2^e) = 2^{e-1}$, $2^{e-1} \mid \varphi(n)$ and therefore $2^{e-1} \mid r$. Thus, by Euler's theorem, $x^r \equiv_{2^e} 1$ for *all* $2^{e-1}$ distinct $x \in \mathbf{Z}^*_{\mathbf{2^e}}$. Thus, there are $2^{e-1} = \varphi(2^e)$ distinct $r^{\text{th}}$ roots of 1 modulo $2^e$, and since $2^{e-1} = \varphi(2^e)$ divides $r$, there are $\gcd(\varphi(2^e), r)$ distinct $r^{\text{th}}$ roots of 1 modulo $2^e$.

When $r$ is odd, $x^r \equiv_{2^e} 1$ implies that $x^r - 1 \equiv_{2^e} 0$ and therefore that $2^e \mid (x^r - 1)$. But $(x^r - 1) = (x-1)(x^{r-1} + x^{r-2} + \cdots + x + 1)$, and the right hand term contains $r$ summands each of which is odd (since all elements of $\mathbf{Z}^*_{\mathbf{2^e}}$ are odd). Thus, the right hand term is odd. This implies that $2^e \mid (x-1)$ and hence that $x \equiv_{2^e} 1$ is the only $r^{\text{th}}$ root of 1 modulo $2^e$. Therefore, there are once again $\gcd(\varphi(2^e), r) = 1$ distinct $r^{\text{th}}$ roots of 1 modulo $2^e$. Thus, for any prime $p_i$, the number of distinct $r^{\text{th}}$ roots of 1 is exactly $h_i$.

By the Chinese Remainder Theorem, since all of the $p_i^{e_i}$ are relatively prime, the total number of $r^{\text{th}}$ roots of 1 modulo $n$ is $H = \prod_i h_i$. Now write $r = q_1 q_2 \cdots q_\ell$ where the $q_j$ are *not necessarily distinct* primes. Since each prime $q_j \mid r$, since $r \mid \varphi(n)$, and since $\varphi(n) = \prod_i \varphi(p_i^{e_i})$, then each prime $q_j \mid \varphi(p_i^{e_i})$ for some $i$. Since $r = \prod_j q_j$ and $\varphi(n)/r = \varphi(n)/(\prod_j q_j)$ are relatively prime, each $q_j$ can be associated with an $h_i$ such that $r = \prod_j q_j = \prod_i h_i = H$. Thus, there are exactly $r$ distinct $r^{\text{th}}$ roots of 1 modulo $n$.

Finally, by lemma 2.2, there are exactly $r$ distinct $r^{\text{th}}$ roots of $z$ for each $z \in \mathbf{Z^r_n}$. ∎

**Corollary 2.21** *If $r \mid \varphi(n)$ and if $r$ and $\varphi(n)/r$ are relatively prime, then $|\mathbf{Z^r_n}| = \varphi(n)/r$.*

*Proof:*

$|\mathbf{Z}_\mathbf{n}^*| = \varphi(n)$ and every $x \in \mathbf{Z}_\mathbf{n}^*$ is an $r^{\text{th}}$ root of some $z \in \mathbf{Z}_\mathbf{n}^\mathbf{r}$ (namely $z \equiv_n x^r$). Since, by theorem 2.20, every $z \in \mathbf{Z}_\mathbf{n}^\mathbf{r}$ has exactly $r$ distinct $r^{\text{th}}$ roots, $|\mathbf{Z}_\mathbf{n}^\mathbf{r}| = \varphi(n)/r$, as desired. ∎

**Definition** The triple $(r, n, y)$ is said to be *perfect consonant* if $(r, n, y)$ is consonant and if $r$ and $\varphi(n)/r$ are relatively prime.

The following theorem characterizes the cosets of $\mathbf{Z}_\mathbf{n}^\mathbf{r}$ in $\mathbf{Z}_\mathbf{n}^*$ when $(r, n, y)$ is perfect consonant.

**Theorem 2.22** *Let $(r, n, y)$ be a perfect consonant triple and let $w \in \mathbf{Z}_\mathbf{n}^*$. $w$ is expressible as $w \equiv_n y^c z$ for a unique integer $c$ in the range $0 \leq c < r$ and a unique $z \in \mathbf{Z}_\mathbf{n}^\mathbf{r}$.*

*Proof:*

A simple counting argument suffices here. By corollary 2.21, $|\mathbf{Z}_\mathbf{n}^\mathbf{r}| = \varphi(n)/r$, and since there are $r$ integers $c$ in the range $0 \leq c < r$, there are exactly $\varphi(n)$ pairs $(c, z)$. Since for every pair $(c, z)$, $y^c z \in \mathcal{RC}[c]$ (by definition), and since (by theorem 2.17) no two distinct pairs correspond to the same $y^c z \in \mathbf{Z}_\mathbf{n}^*$, there are at least $\varphi(n)$ distinct $y^c z \in \mathbf{Z}_\mathbf{n}^*$. But $|\mathbf{Z}_\mathbf{n}^*| = \varphi(n)$. Thus for every $w \in \mathbf{Z}_\mathbf{n}^*$, there exists *exactly* one pair $(c, z)$ with $0 \leq c < r$ and $z \in \mathbf{Z}_\mathbf{n}^\mathbf{r}$ such that $w \equiv_n y^c z$. ∎

Therefore, when $(r, n, y)$ is perfect consonant, the residue classes

$$\mathcal{RC}[0], \mathcal{RC}[1], \ldots, \mathcal{RC}[r-1]$$

form a partition of $\mathbf{Z}_\mathbf{n}^*$.

## 2.5 Deciding Residue Classes

Theorem 2.22 shows that when $(r, n, y)$ is perfect consonant, then *every* $w \in \mathbf{Z}_\mathbf{n}^*$ is a member of $\mathcal{RC}[c]$ for *exactly* one integer $c$ in the range $0 \leq c < r$.

**Definition** Let $(r, n, y)$ be a triple of integers. For each $w \in \mathbf{Z}_\mathbf{n}^*$, define $[\![w]\!]_{(r,n,y)}$ (or simply $[\![w]\!]$ when $(r, n, y)$ is understood) to be the smallest non-negative integer $c$ such that $w$ is expressible as $w \equiv_n y^c z$, for some $z \in \mathbf{Z}_\mathbf{n}^\mathbf{r}$. When there is no such $c$, $[\![w]\!] = \infty$.

By theorem 2.17, when $(r, n, y)$ is consonant, $[\![w]\!]_{(r,n,y)}$ is the unique $c$ such that $0 \leq c < r$ and $w \in \mathcal{RC}[c]$ when such a $c$ exists and is $\infty$ otherwise. By theorem 2.22, when $(r, n, y)$ is perfect consonant and $w \in \mathbf{Z}_{\mathbf{n}}^*$, then $[\![w]\!]_{(r,n,y)}$ is always defined.

In the case that $(r, n, y)$ is perfect consonant, we would like to be able to, when given a $w \in \mathbf{Z}_{\mathbf{n}}^*$, determine $[\![w]\!]_{(r,n,y)}$. This will be shown to be efficiently computable when $\sqrt{r}$ is of moderate size and when the factorization of $n$ (and therefore the value of $\varphi(n)$) is known.

We begin this task with the following lemma.

**Lemma 2.23** *If $r \mid \varphi(n)$, $r$ and $\varphi(n)/r$ are relatively prime, and $w \in \mathbf{Z}_{\mathbf{n}}^*$, then $w \in \mathbf{Z}_{\mathbf{n}}^{\mathbf{r}}$ if and only if $w^{\varphi(n)/r} \equiv_n 1$.*

*Proof:*

If $w \in \mathbf{Z}_{\mathbf{n}}^{\mathbf{r}}$ then there exists an $x \in \mathbf{Z}_{\mathbf{n}}^*$ such that $w \equiv_n x^r$. Thus, $w^{\varphi(n)/r} \equiv_n (x^r)^{\varphi(n)/r} \equiv_n x^{\varphi(n)} \equiv_n 1$ by Euler's theorem.

Conversely, if $w^{\varphi(n)/r} \equiv_n 1$, then since $r$ and $\varphi(n)/r$ are relatively prime, there exist integers $A$ and $B$ such that $Ar + B\varphi(n)/r = 1$. Thus,

$$w \equiv_n w^{Ar+B\varphi(n)/r} \equiv_n (w^r)^A (w^{\varphi(n)/r})^B \equiv_n (w^r)^A 1^B \in \mathbf{Z}_{\mathbf{n}}^{\mathbf{r}}$$

as desired. ∎

Therefore, given a $w \in \mathbf{Z}_{\mathbf{n}}^*$, one can effectively determine the residue class of $w$ by checking $w, wy^{-1}, wy^{-2}, \ldots$ until a $wy^{-c}$ is found such that $wy^{-c} \in \mathbf{Z}_{\mathbf{n}}^{\mathbf{r}}$. This $c$ will be the $[\![w]\!]$ and will be found after at most $r$ trials since $[\![w]\!]$ is bounded by $r$.

This method is not impractical for moderately sized $r$; however, certain methods may be employed to make this process more efficient.

**Lemma 2.24** *Let $(r, n, y)$ be a perfect consonant triple. Let $w \in \mathbf{Z}_{\mathbf{n}}^*$ and let $A$ and $B$ be integers such that $Ar - B\varphi(n)/r = -1$. Then $\varsigma \equiv_n w^{B\varphi(n)/r}$ is the unique value such that $\varsigma^r \equiv_n 1$ and $[\![\varsigma]\!]_{(r,n,y)} = [\![w]\!]_{(r,n,y)}$.*

*Proof:*

Since $r$ and $\varphi(n)/r$ are relatively prime, there exist integers $A$ and $B$ such that $Ar - B\varphi(n)/r = -1$.

By theorem 2.22, since $w \in \mathbf{Z}_{\mathbf{n}}^*$, $w$ can be written as $w \equiv_n y^c z$ for some $z \in \mathbf{Z}_{\mathbf{n}}^{\mathbf{r}}$; and by lemma 2.23, $z^{\varphi(n)/r} \equiv_n 1$. Thus,

$$\zeta \equiv_n w^{B\varphi(n)/r} \equiv_n y^{cB\varphi(n)/r} z^{B\varphi(n)/r} \equiv_n y^{c(Ar+1)} \equiv_n (y^{cA})^r y^c \equiv_n y^c \overline{z}$$

where $\overline{z} \equiv_n (y^{cA})^r \in \mathbf{Z}_n^r$. Thus, $[\![\zeta]\!] = [\![w]\!]$.

Also,

$$\zeta^r \equiv_n (w^{B\varphi(n)/r})^r \equiv_n w^{B\varphi(n)} \equiv_n 1^B \equiv_n 1.$$

Thus, $\zeta^r \equiv_n 1$ and $\zeta$ is of the same residue class as $w$.

Since, by theorem 2.20, there are exactly $r$ integers $\zeta_i$ which are $r^{\text{th}}$ roots of 1 modulo $n$; and since, by the above, there is a $\zeta_i$ in each of the $r$ distinct residue classes. Thus, the $\zeta$ of any given residue class must be unique. ∎

**Definition** We call the $\zeta$ of lemma 2.24 the *canonical root of class c.*

If we are to decide the residue class of many distinct integers with respect to a given $r$, $n$, and $y$, lemma 2.24 allows us to, in one preprocessing phase, compute the canonical root of each residue class. Afterwards, we may determine the class of a given $w$ by computing the canonical root $\zeta$ of its class and looking up the class of this $\zeta$ in the precomputed list.

A somewhat more efficient method of determining residue classes can be achieved by using the so called "big step – little step" method.[1] Let $\zeta_*$ be the canonical root of the same class as a given $y$. In $O(\sqrt{r})$ computations, the list $\zeta_1, \zeta_2, \ldots, \zeta_{\lfloor\sqrt{r}\rfloor}$ can be constructed where $\zeta_i \equiv_n \zeta_*^{\lfloor i\sqrt{r}\rfloor}$. Thus, by taking "big" steps, markers can be placed throughout the cyclic group of $r^{\text{th}}$ roots of 1 modulo $n$. Once the list is constructed, a given $\zeta$ can be "looked up" by computing the sequence $\zeta, \zeta\zeta_*, \zeta\zeta_*^2, \ldots$ until a $\zeta\zeta_*^j$ is found which is on the previous list. Such an entry will be found by the time $j$ reaches $j = \lfloor\sqrt{r}\rfloor$. Hence, it takes $O(\sqrt{r}\log r)$ modular operations to compute and sort the initial list and $O(\sqrt{r}\log r)$ modular operations to compute and search the list to find the residue class of a given element.

The reader may observe the similarity between the problem of determining the residue class of a canonical root $\zeta$ and that of evaluating discrete logarithms modulo a prime. Techniques for the latter problem can be applied directly to the former. (See [PoHe78], [Adle79], [COS86] for work on the discrete logarithm problem.)

[1] Thanks to Joe Kilian for pointing out this trick.

One final point to be made here is that if $r$, rather than being prime, is a product of small primes (in particular, $r = 3^k$), then the class of a given $w$ with respect to a given $r$, $n$, and $y$ can be computed in time polynomial in $\log r$. This can be done by (assuming $r = 3^k$) first deciding which of $w$, $wy^{-1}$, and $wy^{-2}$ is a cube ($3^{\text{rd}}$ residue) modulo $n$. Let $i_i \in \{0,1,2\}$ be the (unique) integer such that $wy^{-i}$ is a cube. Let $i_2 \in \{0,1,2\}$ be the (unique) integer such that $wy^{-3i_2-i_1}$ is a $9^{\text{th}}$ residue. Continue for $k$ iterations. It is not hard to see that each choice taken gives a digit in the ternary representation of $[\![w]\!]$.

## 2.6 Evaluations and Decompositions

**Definition** For a given $r$, let $\mathbf{Z_r} = \{\text{integers } c : 0 \leq c < r\}$ denote the set of non-negative integers less than $r$. For $y \in \mathbf{Z_n^*}$, define $\mathbf{D^r_{(n,y)}} = \mathbf{Z_r} \times \mathbf{Z_n^*}$ be the product group with the following somewhat unusual operation $\oplus$:

$$[c,x] \in \mathbf{D^r_{(n,y)}} \qquad \text{if and only if} \qquad c \in \mathbf{Z_r} \quad \text{and} \quad x \in \mathbf{Z_n^*}$$

$$[c_1,x_1] \oplus [c_2,x_2] = \begin{cases} [c_1+c_2, x_1x_2 \bmod n], & \text{if } c_1+c_2 < r; \\ [c_1+c_2-r, x_1x_2y \bmod n], & \text{if } c_1+c_2 \geq r. \end{cases}$$

The rational for this rather bizarre operation shall become clear shortly. First, however, we shall see that $\mathbf{D^r_{(n,y)}}$ is a group.

**Lemma 2.25** $\mathbf{D^r_{(n,y)}}$ *is a commutative group.*

*Proof:*

It is easy to see that the element $[0,1] \in \mathbf{D^r_{(n,y)}}$ is an identity.

By the definition of the $\oplus$ operator, the first component is in $\mathbf{Z_r}$, and since $y \in \mathbf{Z_n^*}$, the second component is in $\mathbf{Z_n^*}$. Hence, $\mathbf{D^r_{(n,y)}}$ is closed.

It is not hard to see that the inverse of $[c,x]$ (denoted by $\ominus[c,x]$) is given by

$$\ominus[c,x] = \begin{cases} [0, x^{-1} \bmod n], & \text{if } c = 0; \\ [r-c, x^{-1}y^{-1} \bmod n], & \text{if } 0 < c < r. \end{cases}$$

Here, if $c = 0$,

$$[0,x] \oplus (\ominus[0,x]) = [0,x] \oplus [0, x^{-1} \bmod n] = [0,1],$$

and if $0 < c < r$, then

$$[c,x] \oplus (\ominus[c,x]) \;=\; [c,x] \oplus [r-c, x^{-1}y^{-1} \bmod n]$$

$$\begin{aligned} &= [c + (r - c) - r, x(x^{-1}y^{-1})y \bmod n] \\ &= [0, 1]. \end{aligned}$$

To see that $\mathbf{D}^{\mathbf{r}}_{(\mathbf{n},\mathbf{y})}$ is associative, we need only observe that

$$\begin{aligned} &([c_1, x_1] \oplus [c_2, x_2]) \oplus [c_3, x_3] \\ &\quad = [c_1 + c_2 + c_3 \bmod r, x_1x_2x_3y^{\lfloor (c_1+c_2+c_3)/r \rfloor} \bmod n] \\ &\quad = [c_1, x_1] \oplus ([c_2, x_2] \oplus [c_3, x_3]). \end{aligned}$$

Finally, the commutativity of addition and multiplication of integers and the symmetry in the definition of $\oplus$ imply that $\mathbf{D}^{\mathbf{r}}_{(\mathbf{n},\mathbf{y})}$ is commutative. ∎

We can begin to understand why $\mathbf{D}^{\mathbf{r}}_{(\mathbf{n},\mathbf{y})}$ is defined as it is by considering the following evaluation function.

**Definition** Define $\psi : \mathbf{D}^{\mathbf{r}}_{(\mathbf{n},\mathbf{y})} \to \mathbf{Z}^*_{\mathbf{n}}$ by $\psi([c, x]) = y^c x^r \bmod n$.

**Lemma 2.26** *$\psi$ is a homomorphism.*

*Proof:*

Let $[c_1, x_1], [c_2, x_2] \in \mathbf{D}^{\mathbf{r}}_{(\mathbf{n},\mathbf{y})}$.

If $c_1 + c_2 < r$, then

$$\begin{aligned} &\psi([c_1, x_1] \oplus [c_2, x_2]) \\ &\quad = \psi([c_1 + c_2, x_1x_2 \bmod n]) \\ &\quad = y^{c_1+c_2}(x_1x_2)^r \bmod n \\ &\quad = (y^{c_1}x_1^r \bmod n)(y^{c_2}x_2^r \bmod n) \bmod n \\ &\quad = \psi([c_1, x_1])\psi([c_2, x_2]). \end{aligned}$$

If $c_1 + c_2 \geq r$, then

$$\begin{aligned} &\psi([c_1, x_1] \oplus [c_2, x_2]) \\ &\quad = \psi([c_1 + c_2 - r, x_1x_2y \bmod n]) \\ &\quad = y^{c_1+c_2-r}(x_1x_2y)^r \bmod n \\ &\quad = y^{c_1+c_2}(x_1x_2)^r \bmod n \\ &\quad = (y^{c_1}x_1^r \bmod n)(y^{c_2}x_2^r \bmod n) \bmod n \\ &\quad = \psi([c_1, x_1])\psi([c_2, x_2]). \end{aligned}$$

Thus, $\psi$ is a homomorphism. ∎

**Definition** Given $[c,x] \in \mathbf{D}^{\mathbf{r}}_{(\mathbf{n},\mathbf{y})}$, $\psi([c,x])$ is called the *evaluation* of $[c,x]$. Given $w \in \mathbf{Z}^*_{\mathbf{n}}$, a $[c,x]$ such that $\psi([c,x]) = w$ is called a *decomposition* of $w$.

**Remarks**

1. A given $w \in \mathbf{Z}^*_{\mathbf{n}}$ may have multiple decompositions in a given $\mathbf{D}^{\mathbf{r}}_{(\mathbf{n},\mathbf{y})}$. It is clear, however, that when $0 \leq c < r$, then $w$ has a decomposition of the form $[c,x]$ if and only if $w \in \mathcal{RC}[c]$.

2. For a given $r$, $n$, and $y$, it is easy to evaluate $\psi([c,x])$ — even if the factorization of $n$ is unknown. It may, however, be far more difficult to find even a single decomposition of a given $w$. In fact, simply computing $[\![w]\!]$ or determining some $c$ such that $w \in \mathcal{RC}[c]$ may be hard.

3. It should now be somewhat clearer why the awkward operation $\oplus$ is used. If the normalization of the first component were not accompanied by normalization of the second component, then $\psi$ would not be a homomorphism. If neither component were normalized, then it would not be possible to restrict the first component to elements of $\mathbf{Z}_{\mathbf{r}}$, and without this restriction, it is very difficult to define $\mathbf{D}^{\mathbf{r}}_{(\mathbf{n},\mathbf{y})}$ so that $\psi$ is a homomorphism (unless $\varphi(n)$ is used, and it may be desirable to keep $\varphi(n)$ private).

By disclosing a decomposition $[c,x]$ of a given $w$, it is easy to show that $w \in \mathcal{RC}[c]$. It will also be important to be able to show how the residue classes of two given integers relate *without* giving their individual residue classes.

**Definition** For convenience, we define the binary difference $[c_1,x_1] \ominus [c_2,x_2]$ of two elements $[c_1,x_1],[c_2,x_2] \in \mathbf{D}^{\mathbf{r}}_{(\mathbf{n},\mathbf{y})}$ by

$$[c_1,x_1] \ominus [c_2,x_2] = [c_1,x_1] \oplus (\ominus[c_2,x_2])\,.$$

The relevant feature of this operation is that given decompositions $[c_1,x_1]$ of $w_1$ and $[c_2,x_2]$ of $w_2$, one can show the residue class of $w_1 w_2^{-1}$ by giving $[c_1,x_1] \ominus [c_2,x_2]$ which is a decomposition of $w_1 w_2^{-1}$ *without* directly revealing the individual residue classes of $w_1$ and $w_2$.

## 2.7 The Election Encryption Function $\mathcal{E}$

At this point, we may begin the task of defining the election encryption function $\mathcal{E}$.

**Definition** For each positive integer $n$, define the *size* of $n$ by $|n| = \lceil \log_2 n \rceil$.

**Definition** A pair of positive integers $(r, n)$ is said to be *exact consonant* if $r$ is prime and if $n$ is the product of two distinct primes $n = pq$ with $|p| = |q| = \lceil |n|/2 \rceil$ such that $r \mid (p-1)$, $r^2 \nmid (p-1)$, and $r \nmid (q-1)$.

**Definition** A triple of positive integers $(r, n, y)$ is said to be *exact consonant* if the pair $(r, n)$ is exact consonant and if $y \in \overline{\mathbf{Z}_n^r}$. Note that an exact consonant triple is both prime consonant and perfect consonant.

We would now like to show that exact consonant pairs and triples are plentiful, easy to find, and can be generated uniformly.

**Lemma 2.27** *Given any odd prime $r$, an exact consonant pair $(r, n)$ with $|n| = N$ together with the factors of $n$ can be selected uniformly in expected time polynomial in $N$.*

(Note that there exist no exact consonant triples $(r, n, y)$ with $r = 2$.)

*Proof:*

We begin by noting that there exist practical random tests to determine whether or not a given integer is prime. The probabilistic primality tests of Solovay and Strassen ([SoSt77]) and Miller ([Mill76]) are quite practical and well suited for these purposes and run in time polynomial in the length of the prospective prime. The probability that these algorithms will, when given a composite, assert that the integer is prime is exponentially small in the number of polynomial time iterations performed. Newer primality tests which never give a "false" prime have been developed by Goldwasser and Kilian ([GoKi86]) and Adleman and Huang ([AdHu87]). These tests run in expected polynomial time in the length of the prospective prime, but they are far less practical than the earlier methods.

The prime number theorem asserts that for $x > 100$, the number of primes less than $x$ is at least $x/\log_e x$. A generalization of the prime number theorem asserts that in any arithmetic sequence $ax + b$ with $x$ a positive integer and $a$ and $b$ relatively prime, the density of primes within this sequence is roughly the same as within the integers. More precisely, the probability that an element

of size $|p|$ randomly chosen from within this sequence is prime is greater than $p/\log_e p$ (see [Kran86]).

An integer $q$ with $|q| = N$ such that $r \nmid (q-1)$ can be uniformly chosen by randomly selecting an integer $b \neq 1$ with $0 \leq b < r$ and selecting $q$ of size $N$ from the sequence $rx + b$. Similarly, an integer $p$ with $|p| = N$ such that $r \mid (p-1)$ and $r^2 \nmid (p-1)$ can be uniformly chosen by randomly selecting an integer $b$ with $0 < b < r$ and selecting $p$ of size $N$ from the sequence $r^2x + br + 1$.

An exact consonant pair can therefore be generated by choosing $p$ and $q$ as prescribed above and checking for primality. If $p$ and $q$ are not prime, they are discarded and another pair is chosen. The general form of the prime number theorem ensures that at least a $1/N$ fraction of such $p$ and of such $q$ are prime. Since the selection and testing is polynomial time and the number of pairs which must be examined is expected to be polynomial in $N$, this process requires expected polynomial time in $N$. ∎

**Lemma 2.28** *Given any exact consonant pair $(r, n)$ together with the factors of $n$, an exact consonant triple $(r, n, y)$ can be selected uniformly in expected time polynomial in $N = |n|$.*

*Proof:*

When $(r, n)$ is exact consonant, by corollary 2.21, precisely 1 out of $r$ of the members of $\mathbf{Z}_n^*$ are $r^{\text{th}}$ residues. Therefore uniform selection of $y$ from $\mathbf{Z}_n^*$ will yield a $y$ which is not an $r^{\text{th}}$ residue with probability $(r-1)/r$. Whether or not a given $y$ is an $r^{\text{th}}$ residue can be quickly tested (given the factors of $n$) by lemma 2.23. Thus, exact consonant triples can be uniformly selected in expected polynomial time in $N$ by uniformly selecting among possible values of $y$ until one is found which is not in $\mathbf{Z}_n^r$. The expected number of trials to find such a $y$ is just $r/(r-1)$. ∎

Note that lemmas 2.27 and 2.28 *do not* together imply that, for fixed $r$, an exact consonant triple $(r, n, y)$ can be selected uniformly from among those with $|n| = N$. This is because the number of exact consonant triples associated with a given exact consonant pair $(r, n)$ is directly proportional to $n$. Thus, even among the exact consonant pairs $(r, n)$ with $|n| = N$, larger values of $n$ will yield more exact consonant triples. Thus for fixed $r$, uniform selection of $n$ followed by uniform selection of $y$ does *not* give uniform selection of an consonant triple

$(r, n, y)$. A weighted selection scheme could yield such a uniform selection process. Fortunately, however, the process of first uniformly selecting $n$ such that $(r, n)$ is exact consonant and then uniformly selecting $y$ such that $(r, n, y)$ is exact consonant will be suitable for the purposes given here.

We are now ready to define an election encryption function.

**Definition** Given an exact consonant triple $(r, n, y)$, an election encryption function $\mathcal{E}_{(r,n,y)}$ is defined to take a secret value $s$ and a random value $x$ and produce $\mathcal{E}_{(r,n,y)}(s, x) = y^s x^r \bmod n$. When there is no ambiguity, the $(r, n, y)$ will be omitted.

Note that where $\mathcal{E}$ is defined, $\mathcal{E}(s, x) = \psi([s, x])$.

It is clear from lemma 2.7 that if $x$ is selected uniformly from $\mathbf{Z}_n^*$, then $\mathcal{E}(s, x)$ is a uniformly selected member of $\mathcal{RC}[s]$. Thus, $\mathcal{E}(s, x)$ serves as an encryption of the secret value $s$ which can be decrypted by any agent which possesses the factorization of $n$ (see section 2.5).

This encryption method serves as a means of probabilistic encryption which is very similar to that given in [GoMi84]. However, since each encrypted value represents the encryption of a message from a message space of size $r$ (instead of size 2 as in the [GoMi84] scheme), the size of a message which is represented by a single encryption can be much larger, and therefore the ratio of plaintext to ciphertext is much larger. In fact, for a given security parameter $N$, the density in the [GoMi84] scheme is only $1/N$ while in this scheme it is $(\log r)/(N + \log r)$ and thus can be made arbitrarily close to 1 by increasing $r$.

This approach is described in the appendix of [BeYu86].

## 2.8 The Prime Residuosity Assumption

Section 2.7 describes how values may be encrypted using an election encryption function $\mathcal{E}$, and section 2.5 shows how such an encryption function can be decrypted when the factorization of the associated $n$ is known. We want it to be the case that it is difficult to decrypt an election encryption function when the factorization of its $n$ is unknown.

Since the problem of distinguishing between residue classes, together with all problems which have been used as the basis for public-key cryptography, are in the class of $\mathcal{NP}$-functions, no non-trivial lower bounds on their compu-

tational complexity are known. The best that we are able to do is to make an assumption about the difficulty of some mathematical problem and prove theorems relative to this assumption. The assumption used here is the *Prime Residuosity Assumption.*

**Definition** Let $\mathcal{A}$ be a (possibly probabilistic) algorithm with input $s$ and binary output. Let $\mathbf{S_0}$ and $\mathbf{S_1}$ be sets and let $p_i$ be the probability that output$(\mathcal{A}) = 1$ given that $s \in \mathbf{S_i}$. $\mathcal{A}$ is said to *distinguish* between $\mathbf{S_0}$ and $\mathbf{S_1}$ *with* $\varepsilon$ *advantage* if

$$|p_1 - p_0| > \varepsilon.$$

$\mathcal{A}$ is said to *distinguish* between $\mathbf{S_0}$ and $\mathbf{S_1}$ *with confidence* $1 - \delta$ if $\mathcal{A}$ distinguishes between $\mathbf{S_0}$ and $\mathbf{S_1}$ with $1 - \delta$ advantage.

The terms "advantage" and "confidence" are technically interchangeable. They do, however, have different connotations. The term "$\varepsilon$ advantage" will be used when $\varepsilon$ is presumed to be near 0 and indicates a slight advantage at distinguishing between sets. The term "confidence $1 - \delta$" will be used when $\delta$ is near 0 and indicates a large advantage at distinguishing between sets.

**Definition** A (possibly probabilistic) algorithm $\mathcal{A}$ is said to *decide* $r^{\text{th}}$ *residues* in polynomial time if there exists some polynomial $P$ and infinitely many positive integers $N$ such that $\mathcal{A}$ distinguishes between $\mathbf{Z_n^r}$ and $\mathbf{Z_n^*}$ with a $1/P(N)$ advantage for at least $1/P(N)$ of the exact consonant pairs $(r, n)$ with $N = |n|$ in time bounded by $P(N)$.

### The Prime Residuosity Assumption

The prime residuosity assumption asserts that for every prime $r$, there exists no probabilistic or deterministic algorithm $\mathcal{A}$ which decides $r^{\text{th}}$ residues in polynomial time.

Although we cannot prove the prime residuosity assumption we can argue about its plausibility. We begin by showing that the ability to distinguish between *some* pair of residue classes implies the ability to distinguish between *every* pair of residue classes.

**Lemma 2.29** *Let $(r, n, y)$ be prime consonant. If one can distinguish between the members of some pair of classes with an $\varepsilon$ advantage in time polynomial in $|n|$, then one can distinguish between the members of any pair of classes with*

*confidence* $1-\delta$ *(i.e. with* $1-\delta$ *advantage) in time polynomial in* $|n|$, $r$, $1/\varepsilon$, *and* $1/\delta$. *In fact, the class of any element can be determined with confidence* $1-\delta$ *in this time.*

*Proof:*

Assume there exists such an algorithm $\mathcal{A}$ which distinguishes between two (possibly unknown) classes with $\varepsilon$ advantage in time polynomial in $|n|$. Let $p_i$ be the probability that output$(\mathcal{A})=1$ when the input to $\mathcal{A}$ is of $\mathcal{RC}[i]$. We begin by extensively sampling the output of $\mathcal{A}$ when given elements chosen uniformly from each of the $r$ possible residue classes. By lemma 2.7, we can uniformly select the elements of any given class, so this is not a problem.

Suppose that we run $\mathcal{A}$ on $k$ uniformly selected elements of a given class $\mathcal{RC}[i]$. The Chebyshev inequality implies that for a given $\varepsilon$, the probability that the observed fraction of trials for which output$(\mathcal{A})=1$ differs $p_i$ by more than $\varepsilon$ is less than $\frac{1}{4k\varepsilon^2}$. Thus, if we sample the output of $\mathcal{A}$ on $k$ randomly selected elements of $\mathcal{RC}[i]$, we can determine $p_i$ to within $\frac{\varepsilon}{4r}$ with a probability of error less than $\frac{4r^2}{k\varepsilon^2}$. Let $Q_i$ be the number of times output$(\mathcal{A})=1$ when given an element of class $i$, and let $q_i=\frac{Q_i}{k}$. That is, $q_i$ is the observed probability that output$(\mathcal{A})=1$ when given an element of class $i$. Thus, $|q_i-p_i|<\frac{\varepsilon}{4r}$ with probability greater than $1-\frac{4r^2}{k\varepsilon^2}$.

By repeating this process for each class, we can construct an entire table of $\mathcal{A}$'s performance on each residue class. We say that such a table is *correct* if for each $q_i$ in the table, $|q_i-p_i|<\frac{\varepsilon}{4r}$. Since each of the individual entries exceeds this bound with probability less than $\frac{4r^2}{k\varepsilon^2}$, $k$ trials on each of the $r$ residue classes produces a table which is correct with probability greater than $1-\frac{4r^3}{k\varepsilon^2}$.

By assumption, there exists at least one pair of classes $i$ and $j$ which is distinguished by $\mathcal{A}$ with $\varepsilon$ advantage. That is, for at least one pair $i,j$, $|p_i-p_j|>\varepsilon$. Thus, given a correct table, there exists at least one pair of entries $(q_i,q_j)$ such that $|q_i-q_j|>\varepsilon-\frac{\varepsilon}{2r}$. Fix $i$ and $j$ such that $q_i$ and $q_j$ are one such pair.

Since $|q_i-q_j|>\varepsilon-\frac{\varepsilon}{2r}$, and since the table contains a total of $r$ values, there must be some interval $I$ of size $\frac{\varepsilon}{r}$ defined by $I=[m-\frac{\varepsilon}{2r},m+\frac{\varepsilon}{2r}]$ with $m$ between $q_i$ and $q_j$ such that no entry of the table falls in $I$. Mark all of the table entries which are below $I$ (less than $m$) with a 0. Mark all of the table entries which are above $I$ (greater than $m$) with a 1.

An integer $w$ whose class $c$ is unknown will be tested as follows. Since $\mathbf{Z}_n^r$ is a group, lemma 2.7 ensures that given an element $w\in\mathcal{RC}[c]$, an element $z$

uniformly selected from $\mathcal{RC}[c]$ can be generated by selecting a random $x \in \mathbf{Z}_n^*$ and forming $wx^r \bmod n$. Similarly (since $w \in \mathcal{RC}[c]$) an element of $\mathcal{RC}[c+i]$ can be selected uniformly by selecting a random $x \in \mathbf{Z}_n^*$ and forming $wy^i z^r \bmod n$. A statistical "fingerprint" of $w$ will be produced by sampling $k$ elements from each of $\mathcal{RC}[c], \mathcal{RC}[c+1], \mathcal{RC}[c+2], \ldots, \mathcal{RC}[c+(r-1)]$. A new set of observed probabilities $q'_\ell$ is obtained. This fingerprint is said to be correct if none of these values differs from the associated actual probability $p_\ell$ by more than $\frac{\varepsilon}{4r}$. (Note that the values $p_\ell$ are simply a reordering — actually just a rotation — of the original probabilities $p_i$, while the $q'_\ell$ represent a different set of observations and therefore may have no relationship to the original $q_i$.) The probability, for a given $\ell$, that $|p_\ell - q'_\ell| > \frac{\varepsilon}{4r}$ is less than $\frac{4r^2}{k\varepsilon^2}$. Thus, the probability that the finger print is correct is greater than $1 - \frac{4r^3}{k\varepsilon^2}$. Each entry of $w$'s fingerprint which is less than $m$ is marked with a 0, and each entry which is greater than $m$ is marked with a 1.

When given a correct table and a correct fingerprint, neither the table entry nor the fingerprint entry for a given pair will vary from the actual value by more than $\frac{\varepsilon}{4r}$. Hence, they will not vary from each other by more than $\frac{\varepsilon}{2r}$. Thus, since the interval $I = [m - \frac{\varepsilon}{2r}, m + \frac{\varepsilon}{2r}]$ contains no table entries, no entry from the fingerprint of $w$ will "span" $I$ and be marked with a 0 (resp. 1) when the corresponding table entry is marked with a 1 (resp. 0).

Now, if the table and fingerprint are correct, there is exactly one rotation of the fingerprint such that its marks will match those of the table. There is at least one matching because rotating the fingerprint by $c$ positions causes the classes (and hence the markings) to match. If there were some other matching (say a rotation of $c'$ positions), then the markings would repeat every $g = \gcd(r, |c - c'|)$ positions. But $r$ is prime, thus either $c \equiv_r c'$ (giving a unique residue class $c$), or $g = 1$. This latter case would imply that the marking sequence is constant (since it repeats every $g = 1$ positions). But this is impossible since it contains (by construction) a 0 in either the $i$ or the $j$ position and a 1 in the other. Hence, this process determines $c$ accurately when the table and fingerprint are correct, and this will occur with probability at least $1 - \frac{8r^3}{k\varepsilon^2}$.

To ensure that the confidence is at least $1 - \delta$, we simply choose $k$ such that $\delta \geq \frac{8r^3}{k\varepsilon^2}$. This is true when $k \geq \frac{8r^3}{\delta\varepsilon^2}$. Thus, since (by assumption) each of the $2kr$ trials can be performed in time polynomial in $|n|$, the entire process can be completed in time polynomial in $|n|$, $r$, $1/\varepsilon$, and $1/\delta$, as desired. ∎

In particular, an algorithm which can gain an inverse polynomial advantage at distinguishing between some pair of residue classes can be made to gain an inverse polynomial advantage at distinguishing between every pair of residue classes.

**Corollary 2.30** *Let $r$ be a fixed prime, and let $P$ be a polynomial. Let $\mathcal{A}$ be an algorithm which for some given prime consonant triples $(r, n, y)$ distinguishes between the members of some pair of classes (with respect to $r$, $n$, and $y$) with an advantage of at least $1/P(|n|)$ in time polynomial in $|n|$. On those same prime consonant triples, one can distinguish between the members of any pair of classes with confidence $1 - \delta$ in time polynomial in $|n|$, $r$, and $1/\delta$. In particular, $\mathbf{Z}_n^r$ and $\mathbf{Z}_n^*$ can be distinguished with a $1/P(|n|)$ advantage in this time.*

*Proof:*

Apply lemma 2.29 with $\varepsilon = 1/P(N)$. ∎

Adleman and McDonnell in [AdMc82] (see also [APR83], [Adle80]) show that an oracle which takes an $r$ and a $z$ and determines whether or not $z$ is an $r^{\text{th}}$ residue (modulo $n$) can be used to generate an efficient (although not quite polynomial time) algorithm to factor $n$. The problem of deciding $r^{\text{th}}$ residuosity (for a fixed $r$) is certainly no harder than factoring. The result of Adleman and McDonnell leads us to believe that the problems may be of comparable complexity.

When $n$ is the product of two distinct primes, computing $\varphi(n)$ is computationally equivalent to factoring $n$. Miller shows in [Mill75] that for general $n$, the two problems are polynomial time equivalent (assuming the Extended Riemann Hypothesis).

One further lemma helps to argue that the special conditions of $r^{\text{th}}$ residuosity and exact consonance are closely related to other problems. As in the quadratic case, the ability to extract roots is equivalent to the ability to split $n$ as shown by lemma 2.31.

**Lemma 2.31** *If there exists some prime divisor $p$ of $n$ such that $r$ and $\varphi(p) = (p-1)$ are relatively prime, and if $x_1^r \equiv_n x_2^r$ for distinct $x_1, x_2 \in \mathbf{Z}_n^*$, then $\gcd(x_1 - x_2, n)$ is a non-trivial factor of $n$.*

*Proof:*

Since $x_1^r \equiv_n x_2^r$, $x_1^r - x_2^r \equiv_n 0$. Since $x_1^r \equiv_n x_2^r$ and $p \mid n$, $x_1^r \equiv_p x_2^r$, and therefore, by lemma 2.3, $x_1 \equiv_p x_2$. $x_1$ and $x_2$ are, however, distinct modulo $n$. Hence, $p \mid (x_1 - x_2)$ and $n \nmid (x_1 - x_2)$. Thus, $\gcd(x_1 - x_2, n)$ is a non-trivial divisor of $n$. ∎

In particular, when $(r, n, y)$ is prime consonant, the ability to either compute $\varphi(n)$ or to extract $r^{\text{th}}$ roots is computationally equivalent to the ability to factor $n$.

## 2.9 Elementary Elections

We are now ready to describe an oversimplified method by which verifiable secret-ballot elections can be held. The scheme presented here is riddled with cracks and should not be read as a claim of a secure method of holding verifiable secret-ballot elections. Instead, the aim of this section is to provide an overview of the method of elections that will be described in greater detail in subsequent chapters.

The scheme described in this section will be centralized. A central *government* is assumed to exist. The government prepares an election encryption function $\mathcal{E}$ as described in section 2.7 with $r$ greater than the number of eligible voters. The government releases to the public the triple $(r, n, y)$ allowing everyone to compute $\mathcal{E}$ of any chosen values, but keeps secret the factors of $n$.

Each voter selects a random $x_i \in \mathbf{Z}_n^*$ and a vote $s_i$. When $s_i = 0$ it denotes a "no vote" and when $s_i = 1$ it denotes a "yes vote". Each voter then publically releases the result $w_i = \mathcal{E}(s_i, x_i)$. By lemma 2.7, $w_i$ will be a uniformly chosen member of $\mathcal{RC}[0]$ for those voters who cast no votes and a uniformly chosen member of $\mathcal{RC}[1]$ for those voters who cast yes votes.

By lemma 2.8, the product $W = \prod_i w_i \bmod n$ will be a member of a $\mathcal{RC}[c]$ where $c$ is equivalent modulo $r$ to the sum of the classes of the released votes $w_i$. Furthermore, by theorem 2.17, this $c$ is unique in the range $0 \leq c < r$. But the number of voters is less than $r$, and each vote is either of class 0 or of class 1. Thus, the sum of these classes, $c$, is precisely equal to the number of yes votes among the votes cast.

The government can compute $c = [\![W]\!]$, (recall that $W$ is public) as in

section 2.5 and prove to the voters that $W \in \mathcal{RC}[c]$ by releasing a $c$ and $x$ such that $W \equiv_n y^c x^r$ where $x$ is an $r^{\text{th}}$ root of $Wy^{-c}$ computed as in lemma 2.19.

The government thereby convinces all participants and observers that the $c$ it releases is, in fact, the number of yes votes cast, and is therefore the *tally* of the election.

There are, of course, many problems with this scheme. These include, but are not limited to the following.

- There is no reason to believe that the government chose its election encryption function as specified in section 2.7.

  In fact, if $r$ and $\varphi(n)$ are chosen to be relatively prime, then every $W \in \mathbf{Z}_n^*$ can be expressed as $W \equiv_n y^c x^r$ for any $c$ of the government's choosing. Thus, the government can claim any tally it desires.

- There is no reason to believe that the votes $(w_i)$ cast by the voters are chosen only from among elements of $\mathcal{RC}[0]$ or $\mathcal{RC}[1]$.

  If a single voter casts as its vote a $w_i \in \mathcal{RC}[1,000,000]$, for instance, then this one vote increments the tally of the election by 1,000,000. Of course the government could detect this digression, but it may not wish to reveal it.

- The government may, even if honest, unwittingly reveal information which may allow some voters to determine the votes of others.

  For example, the schema described above requires the government to reveal an $r^{\text{th}}$ root of a quantity with which it is presented. It may be possible for one or more voters to force the product $W$ towards a value for which a root of $Wy^{-c}$ is already known (and thus, a distinct root would allow the factorization of $n$ to be determined). In this schema, it is not difficult for the last voter of an election to direct things this way.

- The government knows how every voter voted.

  This information could be revealed to others or used in an improper manner. It is certainly undesirable for any agent to know the private information of others.

- The government may halt the election at any time.

If the government sees that the tally of an otherwise proper election is not to its liking, it may refuse to reveal $[\![W]\!]$. Thus, the tally of the election remains unknown to its participants.

The first three problems described here will be addressed by the methods of chapter 3. The final two problems will be managed by the techniques of chapter 4. Chapter 5 will tie these approaches together and give a complete fault-tolerant schema for holding verifiable secret-ballot elections. The schema is proven to not only allay the concerns expressed above, but also to produce a tally which is correct with extremely high probability. It is also proven that distinguishing between the votes of proper voters who follow their protocols is as difficult a problem as deciding between residue classes. It is thusly shown that the existance of any algorithm which can, even with the assistance of colluding participants, gain even an inverse polynomial advantage at deciding between possible votes among proper voters is sufficient to violate the prime residuosity assumption.

# Chapter 3

# Interactive Proofs and the use of Cryptographic Capsules

This chapter describes a deceptively (almost embarrassingly) simple technique, that of *cryptographic capsules*, which allows Alice to convince Bob that either $X$ or $Y$ is true without giving Bob any information as to which is the case. Capsules are used in and further the capabilities of so called *interactive proofs*.

The notion of *interactive proofs* was introduced and first used by Goldwasser and Micali in [GoMi83]. The ideas were further developed by the original authors together with Fischer and Rackoff in [FMR84] and [GMR85]. The fundamental idea is an extension of the notion of a formal proof to what might be called a convincing argument.

Traditionally, if Alice wants to convince Bob that a certain property holds (say that a given $y$ is a quadratic residue modulo a given $n$) she could write out a formal proof of the fact (which would likely include the factorization of $n$) and send the proof to Bob. This procedure is quite effective at giving Bob the desired information, but it might have an unwanted side effect of giving Bob additional information (such as the factorization of $n$).

Instead, Alice might be able to convince Bob of some property through an interactive conversation—perhaps by answering a number of challenges put forward by Bob. In this way, it may be possible for Alice to convince Bob of the validity of her claim *without* surrendering additional information.

## 3.1 Some Interactive Proofs

Interactive proofs are a kind of electronic shell game. Suppose there is a ball hidden under one of two shells. Alice is allowed to scramble the shells and Bob is then allowed to look beneath one of the two shells. If this process is repeated indefinitely, and if Bob each time randomly selects the shell to look beneath, then Bob will, in all probability, eventually find the ball. In fact, it may be said that Bob will, with extremely high probability, find the ball within a fairly small number of trials. If, after many tries, Bob has not found a ball, he may reasonably conclude that neither shell has a ball beneath it. This is true even though Bob may never have looked beneath both shells simultaneously.

A more substantive example can be drawn from the domain of chapter 2. Suppose that Alice wants to select and reveal to Bob an $n$ such that $n = pq$ where $p$ and $q$ are distinct primes and $|p| = |q|$. Alice may proceed as in figure 3.1.

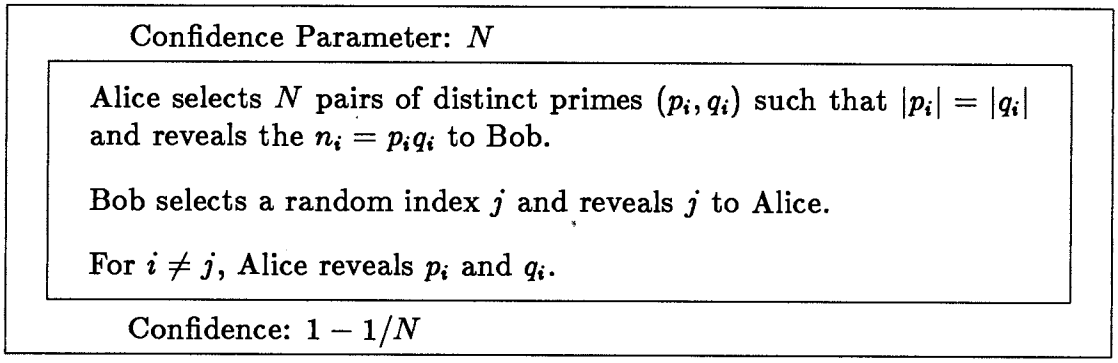


Figure 3.1: Interactive proof that an $n = pq$ where $p$ and $q$ are distinct primes such that $|p| = |q|$.

If for all $i \neq j$, Alice has released $p_i$ and $q_i$ such that $n_i = p_i q_i$, $p_i$ and $q_i$ are prime, and $|p_i| = |q_i|$, then it may be said that Bob is "convinced" that it is also the case that $n_j = p_j q_j$ where $p_j$ and $q_j$ are distinct primes with $|p_j| = |q_j|$.

Alice could only have "defeated" Bob by successfully predicting which $j$ Bob would select. Since Bob selects $j$ randomly, and since there were $N$ possible values of $j$ to choose from, this could only be done with probability $1/N$. Thus, Bob may be said to be "convinced" that Alice could have only cheated him with probability $1/N$. Thus, Bob is said to have "confidence $1 - 1/N$".

Once this interactive proof has been completed, Alice and Bob share a value $n$ for which only Alice possesses the factors but for which Bob is convinced that certain properties hold. It may be claimed that Bob has "learned" nothing from the interactive proof other than the fact that $n$ satisfies these properties. In particular, Bob has not been given the factors of $n$.

The approach used in this interactive proof is very general, and it may be used by Alice in many situations to convince Bob that an object has a given property which is feasibly computable by Alice but not by Bob. There are, however, two principal shortcomings of this approach.

First, although Bob's confidence may be high, it is difficult to make it very high, as is necessary for many applications. The probability that Bob is fooled decreases only in inverse linear proportion to the number of objects prepared by Alice.

Second, it is difficult to use this technique to show that a property holds about a specific object. For instance, it may still be difficult for Alice to convince Bob that a specified $z$ is in $\mathbf{Z}_n^r$ for fixed $n$ and $r$ unless Alice is willing to reveal to Bob the factors of $n$ or an $r^{\text{th}}$ root of the specified $z$.

We shall show here some relevant interactive proofs which remedy these shortcomings. The first is intended to be somewhat introductory.

### 3.1.1 An Interactive Proof that $z \in \mathbf{Z}_n^r$

It is shown in chapter 2 that possession of two distinct $r^{\text{th}}$ roots of a given $r^{\text{th}}$ residue $z$ is often sufficient to split $n$ into non-trivial factors. In particular when $(r, n)$ is exact consonant, lemma 2.31 shows that possession of two distinct $r^{\text{th}}$ roots is sufficient to completely factor $n$. In addition, in such an elementary election scenario of section 2.9, the last voter could direct the outcome so as to force the government to produce an $r^{\text{th}}$ root of any desired $r^{\text{th}}$ residue. Thus, the final voter could select a random $x$ and force the government to release an $r^{\text{th}}$ root $x'$ of $z \equiv_n x^r$. Since $x$ is chosen randomly and not revealed, and since there are $r$ distinct $r^{\text{th}}$ roots of $z$, the probability that $x \not\equiv_n x'$ is $(r-1)/r$. Since $r$ is greater than the number of voters, this probability is quite high. Hence, the final voter could manipulate the election in such a way as to, with high probability, determine the factors of $n$ and thereby determine the votes of other voters.

In order to avoid this and related difficulties, it is desirable that the government have some means of convincing others that a given $z$ is in $\mathbf{Z}_n^r$ *without* revealing an $r^{\text{th}}$ root of $z$ or the factors of $n$. We will limit ourselves here to an intuitive argument as to why this approach does not reveal undue information. The formal proof that privacy is not in any way compromised is deferred to chapter 5.

The essence of the interactive proof of figure 3.2 relies on two properties: first, the group structure of $\mathbf{Z}_n^r$; and second, the fact that random pairs of the form $(x, z)$ where $x$ is an $r^{\text{th}}$ root of $z$ can be easily generated by anyone with or without the factors of $n$.

Confidence Parameter: $N$

Alice prepares $N$ random pairs of the form $(x_i, z_i)$ where each $x_i$ is randomly chosen from $\mathbf{Z}_n^*$ and each $z_i \equiv_n x_i^r$.
Alice reveals the $z_i$ to Bob but keeps the corresponding $x_i$ concealed.

Bob selects a random subset $S$ of the indices.

For each $i \in S$, Alice reveals to Bob $x_i$.
For each $i \notin S$, Alice reveals to Bob $xx_i$ where $x$ is a fixed $r^{\text{th}}$ root of the $z$ in question.

Confidence: $1 - 1/2^N$

Figure 3.2: Interactive proof that $z \in \mathbf{Z}_n^r$.

It can be argued here that Bob has received no information since he has seen only random $(x_i, z_i)$ pairs which he could have generated for himself. It is also the case that it has been proven to Bob that for each $i \in S$, $z_i \in \mathbf{Z}_n^r$; and therefore, since Alice could not have foreseen which subset of indices Bob would choose, Bob can be said to have been "convinced" that $z_i \in \mathbf{Z}_n^r$ for *at least* one $i \notin S$. If this were not the case, then it would have to be true that $z_i \in \mathbf{Z}_n^r$ for every $i \in S$ *and* $z_i \notin \mathbf{Z}_n^r$ for every $i \notin S$. Thus, Alice would have to have either known or guessed the precise subset $S$ to be selected by Bob *before* revealing the $z_i$ to Bob. Alice could guess such a subset of the $N$ indices with probability only $1/2^N$.

In the last step, Alice reveals the $xx_i$ to Bob. Each such $xx_i$ is an $r^{\text{th}}$ root of

$zz_i$, but because of the group structure of $\mathbf{Z}_n^r$, an $r^{\text{th}}$ root of $zz_i$ can only exist if either both $z$ and $z_i$ are in the group $\mathbf{Z}_n^r$ or if neither $z$ nor $z_i$ is in this group. Thus, having been previously convinced that at least one of these latter $z_i$ for $i \notin S$ is in the group $\mathbf{Z}_n^r$, Bob must conclude that $z$ also is in $\mathbf{Z}_n^r$.

The only place where it appears that Bob might gain some undue information from Alice is in this last step. This concern is, however, alleviated by again appealing to the group structure of $\mathbf{Z}_n^r$. Since $\mathbf{Z}_n^r$ is a group, the pairs $(xx_i, zz_i)$ are, to Bob, random elements of $\mathbf{Z}_n^*$ together with their $r^{\text{th}}$ powers. Once again, Bob could have generated such pairs for himself. Thus, it can be argued that, beyond being convinced that $z \in \mathbf{Z}_n^r$, Bob has learned nothing from the exchange.

The limited disclosure of information seen in the interactive proof of figure 3.2 is the essence of "zero-knowledge" as defined by Goldwasser, Micali, and Rackoff in [GMR85]. The notion of zero-knowledge stems from the attempt to capture the intuition that Bob has learned nothing from the interactive proof besides that which is entailed by the claim of the proof itself.

One way in which to capture this intuition is to assert that it must be possible to "simulate" Alice's actions in such a way that Bob's conversations with Alice are indistinguishable from simulated conversations. If this property is true of all possible agents who might take Bob's place in an interactive proof with Alice, then, in some sense, no agent could learn anything from these conversations and the interaction is deemed zero-knowledge.

In the interactive proof of figure 3.2, for example, Bob is convinced of the claim that $z \in \mathbf{Z}_n^r$ as well as of consequences of this claim. It is possible, moreover, to simulate Alice's actions in this protocol, since all of Alice's actions are indistinguishable from the release of randomly chosen pairs of the form $(x, x^r)$.

A simulator could therefore be constructed which first selects the set $S$ of indices exactly as Bob would. Note that a dishonest agent in Bob's place might not select $S$ randomly, but rather use some other means. This does matter as long as $S$ is generated exactly as the agent would. The simulator then randomly generates pairs $(x_i, x_i^r)$ with each $x_i \in \mathbf{Z}_n^*$. For all $i \in S$, the simulator releases $z_i = x_i^r \bmod n$, and for all $i \notin S$, the simulator releases $z_i = z^{-1}x_i^r \bmod r$. The simulator then releases the set $S$. Finally, the simulator releases all $x_i$. By construction, for all $i \in S$, $x_i$ is an $r^{\text{th}}$ root of $z_i$, and for all $i \notin S$, $x_i$ is an $r^{\text{th}}$

root of $zz_i$, as desired.

Thus, no outside observer would be able to distinguish the simulation from an actual conversation between Alice and Bob. This zero-knowledge property ensures, in particular, that Alice is not giving information to Bob which could assist Bob in constructing an $r^{\text{th}}$ root of $z$.

It should once again be emphasized that although this is an appealing arguement, it is not a formal proof. We neither prove that $z \in \mathbf{Z}_{\mathbf{n}}^{\mathbf{r}}$ must be the case nor that Bob has "learned" nothing he shouldn't have in the exchange. In this chapter, the interactive proofs presented will *not* be accompanied by formal proofs that they achieve the desired properties. Instead, arguments like the preceding intuitive appeal will be given. The formal proofs that are required will be given as part of the full verifiable secret-ballot election schema of chapter 5.

### 3.1.2 An Interactive Proof that $w \in \mathcal{RC}[c]$

Given the interactive proof method of section 3.1.1, it is now easy to see how to show that $w \in \mathcal{RC}[c]_{(r,n,y)}$.

By definition, $w \in \mathcal{RC}[c]_{(r,n,y)}$ if and only if there exists some $z \in \mathbf{Z}_{\mathbf{n}}^{\mathbf{r}}$ such that $w \equiv_n y^c z$. Thus, showing that $w \in \mathcal{RC}[c]$ is equivalent to showing that $wy^{-c} \in \mathbf{Z}_{\mathbf{n}}^{\mathbf{r}}$. Hence, $w \in \mathcal{RC}[c]$ can be shown by using the interactive proof method of section 3.1.1 to show that $wy^{-c} \in \mathbf{Z}_{\mathbf{n}}^{\mathbf{r}}$.

In particular, theorem 2.17 tells us that if $(r, n, y)$ is consonant, there exists at most one integer $c$ with $0 \leq c < r$ such that $w$ is expressible as $w \equiv_n y^c z$ for some $z \in \mathbf{Z}_{\mathbf{n}}^{\mathbf{r}}$. This $c$, when defined, is $[\![w]\!]_{(r,n,y)}$. Thus, when $(r, n, y)$ is consonant, this method can be used to show that $[\![w]\!]_{(r,n,y)} = c$.

### 3.1.3 An Interactive Proof that $[\![w_1]\!] = [\![w_2]\!]$

It will soon become important to enable Alice to convince Bob that, *under the assumption that* $(r, n, y)$ *is consonant,* $[\![w_1]\!] = [\![w_2]\!]$ *without* revealing to Bob the values of $[\![w_1]\!]$ and $[\![w_2]\!]$. This capability will be used extensively as a component of subsequent interactive proofs.

Lemma 2.10 tells us that $[\![w_1]\!] = [\![w_2]\!]$ if and only if $w_1 w_2^{-1} \in \mathbf{Z}_{\mathbf{n}}^{\mathbf{r}}$. Thus, an interactive proof that $[\![w_1]\!] = [\![w_2]\!]$ may be achieved by using the interactive proof technique of section 3.1.1 to show that $w_1 w_2^{-1} \in \mathbf{Z}_{\mathbf{n}}^{\mathbf{r}}$.

A uniform indistinguishability argument can be used to show that this interactive proof method gives an adversary no substantial information about the value of $[\![w_1]\!]$ and $[\![w_2]\!]$. There are, however, subtleties to this argument since the phrase "gives an adversary no substantial information" has not yet been precisely defined. Once again, these details will be deferred.

### 3.1.4 An Interactive Proof that $(r, n, y)$ is consonant

We next give an interactive proof method to show that a triple $(r, n, y)$ is consonant. This interactive proof has an added benefit since, by corollary 2.15, when $(r, n, y)$ is consonant and $r > 1$, then $y \notin \mathbf{Z}_{\mathbf{n}}^{\mathbf{r}}$. Thus, the methods of this section can be used in many important cases to show that $y \notin \mathbf{Z}_{\mathbf{n}}^{\mathbf{r}}$. Section 3.3.1 will give an improved interactive proof method which can always be used to show that $y \notin \mathbf{Z}_{\mathbf{n}}^{\mathbf{r}}$.

The key element of this interactive proof is obtained from lemma 2.11 which states that when $y \in \mathbf{Z}_{\mathbf{n}}^{*}$, there are $r$ distinct residue classes precisely when $(r, n, y)$ is consonant and no more than $r/2$ distinct residue classes otherwise. Thus, Alice can convince Bob that $(r, n, y)$ is consonant by demonstrating her ability to distinguish between residue classes. In particular, when Alice is presented with a $w$ randomly chosen from one of the $r$ residue classes $\mathcal{RC}[0], \mathcal{RC}[1], \ldots, \mathcal{RC}[r-1]$, she should always be able to determine to which class $w$ belongs and give a response which matches Bob's expectation.

This suggests the interactive proof that $(r, n, y)$ is consonant given in figure 3.3. Note that if $y \notin \mathbf{Z}_{\mathbf{n}}^{*}$, then by theorem 2.13(a), $(r, n, y)$ is not consonant. This condition is easily tested, so we may assume that $y \in \mathbf{Z}_{\mathbf{n}}^{*}$.

Confidence Parameter: $N$

Bob prepares $N$ pairs $[c_i, x_i]$, where $c_i$ is uniformly chosen from $\mathbf{Z}_{\mathbf{r}}$ and $x_i$ is uniformly chosen from $\mathbf{Z}_{\mathbf{n}}^{*}$, and sends Alice the values $w_i = \psi([c_i, x_i]) = y^{c_i} x_i^r \bmod n$.

Alice computes the $c_i$ from the $w_i$ and sends the $c_i$ to Bob.

Confidence: $1 - 1/2^N$

Figure 3.3: Interactive proof that $(r, n, y)$ is consonant.

Lemma 2.7 guarantees that each $w_i$ is a uniformly selected member of $\mathcal{RC}[c_i]$. Lemmas 2.5 and 2.11 guarantee that if $(r, n, y)$ were not consonant, then $w_i$ would also be a member of $\mathcal{RC}[c_i']$ for some $c_i' \neq c_i$ with $0 \leq c_i' < r$ and that therefore (by lemma 2.4) $\mathcal{RC}[c_i] = \mathcal{RC}[c_i']$. Since $c_i$ was chosen by Bob uniformly from $\mathbf{Z_r}$, Alice could only guess which of $c_i$ or $c_i'$ (or possibly other values) was chosen by Bob. She would therefore have a probability of at most $1/2$ of guessing the correct $c_i$ on each of $N$ trials. Thus, Alice would not be able to duplicate the $c_i$ expected by Bob with probability greater than $1/2^N$.

Despite its simplicity, there is a problem here. Alice is effectively acting as a residue class oracle for Bob and may be giving Bob useful information about the class of one or more $w_i$ for which Bob did not already know the class. Thus, there is no hope that this interactive proof could satisfy the zero-knowledge criterion.

To handle this problem, we add an additional phase to the interactive proof whereby Bob convinces Alice that he already "knows" the class of an element before Alice gives it.

The interactive proof of figure 3.4 can be used by Bob to convince Alice that he "knows" the residue class of a chosen element.

Confidence Parameter: $N$

Bob prepares a pair $[c, x]$ and sends $w = \psi([c, x]) = y^c x^r \bmod n$ to Alice.

Bob also prepares $N$ pairs $[c_i, x_i]$, where $c_i$ is uniformly chosen from $\mathbf{Z_r}$ and $x_i$ is uniformly chosen from $\mathbf{Z_n^*}$, and sends $w_i = \psi([c_i, x_i]) = y^{c_i} x_i^r \bmod n$ to Alice.

Alice selects a random subset $S$ of the indices.

For each $i \in S$, Bob sends $[c_i, x_i]$ to Alice.
For each $i \notin S$, Bob sends Alice $[c, x] \oplus [c_i, x_i]$.

Confidence: $1 - 1/2^N$

Figure 3.4: Interactive proof that a decomposition of $w$ is "known".

For any $i$, if one is given both $[c_i, x_i]$ and $[c, x] \oplus [c_i, x_i]$, then one can easily compute $[c, x]$. Thus, Alice is convinced that Bob can compute $[c, x]$. Because

of the group structure of $\mathbf{D}^{\mathbf{r}}_{(\mathbf{n},\mathbf{y})}$, however, all that Alice sees are $N$ completely random elements of $\mathbf{D}^{\mathbf{r}}_{(\mathbf{n},\mathbf{y})}$. Thus, it can be argued that this interaction is of no help to Alice in attempting to compute $c$.

A complete interactive proof that $(r, n, y)$ is consonant which is satisfactory to both Alice and Bob can now be given in figure 3.5.

Confidence Parameter: $N$

Security Parameter: $N'$

Bob uniformly selects $N$ elements $[c_i, x_i] \in \mathbf{D}^{\mathbf{r}}_{(\mathbf{n},\mathbf{y})}$.
Bob sends Alice $\psi([c_i, x_i])$ for all such $[c_i, x_i]$.
For each $i$, Bob uniformly selects $N'$ additional elements $[c_{i,j}, x_{i,j}] \in \mathbf{D}^{\mathbf{r}}_{(\mathbf{n},\mathbf{y})}$ and sends $\psi([c_{i,j}, x_{i,j}])$ to Alice for all $(i, j)$.

Alice selects a random subset $S$ of the $(i, j)$ indices.

For each $(i, j) \in S$, Bob sends $[c_{i,j}, x_{i,j}]$ to Alice.
For each $(i, j) \notin S$, Bob sends Alice $[c_i, x_i] \oplus [c_{i,j}, x_{i,j}]$.

Alice sends Bob $c_i$ for all $i$.

Confidence: $1 - 1/2^N$

Security: $(1 - 1/2^{N'})^N > 1 - N/2^{N'}$

Figure 3.5: "Secure" interactive proof of consonance.

Notice the notion of "security" added here. Informally, the confidence measure indicates the level of faith of the verifier (Bob) that it has not been deceived by the prover (Alice). The security measure indicates the level of faith of the prover (Alice) that it has not been deceived by the verifier (Bob). Note that the verifier is deceived if it is convinced of a claim which is not true, while the prover is deceived if the verifier is able to gain information beyond that which is entailed by the claim.

In the above interactive proof, Bob could only gain additional information by, in one of $N$ instances, guessing (in advance) a subset of $N'$ elements selected by the Alice. Each subset could be guessed with probability $1/2^{N'}$. Thus, the probability that none of the $N$ subsets is guessed is $(1 - 1/2^{N'})^N$. This gives the "security" level of the interactive proof.

Once again it is emphasized that these notions are *not* meant to be formal. Formal definitions of confidence and security will be given as needed in chapter 5.

**Remark** In most cases where Alice generates a triple $(r, n, y)$ for use in an interactive protocol, it is sufficient that Alice convince Bob that $(r, n, y)$ is consonant. A protocol may specify that Alice select a perfect consonant or even an exact consonant triple. This is generally to enable Alice to easily decide residue classes or to make deciding classes difficult on average for Bob (by making $n$ difficult to factor, for instance). Thus, in these cases, Alice need not convince Bob of perfect consonance or exact consonance, as it is only to her disadvantage to violate these conditions. It is, however, necessary for Alice to convince Bob of consonance since violation of this property may allow Alice to cheat Bob.

## 3.2 Cryptographic Capsules

A *cryptographic capsule* (or simply *capsule*) is a randomly ordered collection of objects, each of which is of some specified form. The order of the elements of the capsule is randomly permuted to hide which element is of which type; or, alternately, some easily computable ordering function (such as $\leq$) can be applied to the capsule to obscure the original ordering.

A simple example of a capsule is a pair of integers — one of which is even and one of which is odd, e.g. $(4, 13)$. This capsule, however, is not very interesting because it is readily apparent which is the odd integer and which is the even integer.

A somewhat more useful capsule may be an (unordered) pair of integers $\{n_1, n_2\}$ with $n_1 = p_1 q_1$ where $p_1$ and $q_1$ are each primes congruent to 1 modulo 4 and $n_2 = p_2 q_2$ where $p_2$ and $q_2$ are each primes congruent to 3 modulo 4.

If we assume that distinguishing between these two cases is hard, then this suggests a simple method for flipping a coin over a telephone. Alice prepares such a pair and transmits it to Bob; Bob then selects one element from the pair and transmits his choice to Alice; finally, Alice reveals the factors of both $n_1$ and $n_2$ to Bob. We may say that the coin flip is heads if Bob chose the element with factors congruent to 1 modulo 4 and tails otherwise.

This is not an ideal example, since Alice could have simply transmitted a single integer of one of the two preceding classes and asked Bob to guess which class it was from. The real power of capsules comes from the ability to prove interactively that a capsule is of the required form without the need to later reveal secret information about its contents.

## 3.3 Capsules and Residues

Many of the interesting uses of cryptographic capsules are found when their contents consist of members of two or more residue classes as described in chapter 2.

### 3.3.1 An Interactive Proof that $y \notin \mathbf{Z}_n^r$

Section 3.1.4 gave an interactive proof method to show that a triple $(r, n, y)$ is consonant. This has the fringe benefit of also showing that $y \notin \mathbf{Z}_n^r$ since this is always the case when $(r, n, y)$ is consonant and $r > 1$. It is, however, not always the case that $y \notin \mathbf{Z}_n^r$ implies that $(r, n, y)$ is consonant. We want, therefore, to give a somewhat more general interactive proof method which can always be used to show that $y \notin \mathbf{Z}_n^r$ whenever it is, in fact, true.

There is another reason for which we would like to reexamine the interactive proof method of section 3.1.4. With that method, Alice must decide the residue class of many distinct values. This may be a cumbersome task for Alice since she has $r$ distinct classes to choose from and even the best methods use $O(\sqrt{r}\log r)$ operations.

The burden on Alice can be eased substantially by limiting the possibilities to two. Alice is presented either with a member of $\mathcal{RC}[0]$ or a member of $\mathcal{RC}[1]$ and asked to decide which is the case. This is sufficient to show that $y \notin \mathbf{Z}_n^r$ since when $y \in \mathbf{Z}_n^r$, $\mathcal{RC}[0] = \mathcal{RC}[1]$.

As in the interactive proof of section 3.1.4, Bob could simply present Alice with randomly chosen members of $\mathcal{RC}[0]$ and $\mathcal{RC}[1]$ and ask her to distinguish between them. Once again, however, by doing this Alice could be acting as a residuosity oracle for Bob and might be giving Bob undue information.

To alleviate this problem, Bob must convince Alice that he "knows" the class of a $w$ before presenting it to Alice *without* giving away the class to Alice.

Furthermore, Bob must prove to Alice that $[\![w]\!]_{(r,n,y)} \in \{0,1\}$, again without giving away which is the case. If Bob fails to do this and presents Alice with a $w$ for which $[\![w]\!] > 1$, then Alice may be forced to search for $[\![w]\!]$ or to go through a cumbersome procedure to accuse Bob of cheating.

To accomplish these tasks, cryptographic capsules can be used as in figure 3.6.

Confidence Parameter: $N$

Bob prepares a pair $[c, x]$ such that $c \in \{0,1\}$ and $x \in \mathbf{Z}_n^*$ and sends $w = \psi([c,x]) = y^c x^r \bmod n$ to Alice.

Bob also selects $N$ additional values $\alpha_i$ and $\beta_i$ and sends to Alice $N$ capsules each consisting of (in random order) the pair $u_i = \psi([0,\alpha_i]) = \alpha_i^r \bmod n$ and $v_i = \psi([1,\beta_i]) = y\beta_i^r \bmod n$.

Alice designates a random subset of the capsules and demands that they be opened.

Bob opens these capsules by presenting Alice with a decomposition of each component of the capsule.

For each remaining capsule, Bob selects

$$\omega_i = \begin{cases} u_i & \text{if } c = 0; \\ v_i & \text{if } c = 1. \end{cases}$$

and shows that $\omega_i$ is of the same residue class as $w$ by presenting Alice with a decomposition of the quotient $\omega_i w^{-1}$.

Confidence: $1 - 1/2^N$

Figure 3.6: Interactive proof that $[\![w]\!] \in \{0,1\}$.

Unless Bob were able to guess, in advance, the subset of capsules which Alice designates to be opened, Bob would not be able to perform the required decompositions without being able to determine the class of $w$ by himself. Bob's chance of guessing such a subset is only $1/2^N$.

The importance of Bob being able to demonstrate that his value is a legitimate member of $\mathcal{RC}[0]$ or $\mathcal{RC}[1]$ without revealing which should not be lost here, as this is one of the fundamentals which will be used later to enable secret-ballot elections to be made verifiable.

Once Bob has completed the interactive proof of figure 3.6, Alice should be convinced that Bob knows $[\![w]\!]$ and that $[\![w]\!] \in \{0,1\}$. Thus, Alice should be willing to provide $[\![w]\!]$ to Bob.

If $\mathcal{RC}[0] = \mathcal{RC}[1]$ were true, then Alice would only have a $1/2$ chance of presenting Bob with the class he expects. Thus, by repeating the entire process $N$ times, Bob's confidence can be elevated to $1 - 1/2^N$. Therefore, both Alice and Bob should be satisfied by the interactive proof of figure 3.7.

Confidence Parameter: $N$

Security Parameter: $N'$

Bob randomly selects $N$ values $c_i \in \{0,1\}$ and $x_i$ uniformly from $\mathbf{Z}_\mathbf{n}^*$ and sends Alice the $N$ values $w_i = \psi([c_i, x_i]) = y^{c_i} x_i^r \bmod n$.

With each $w_i$, Bob randomly selects $N'$ additional values $\alpha_{i,j}$ and $\beta_{i,j}$ and sends to Alice $N'$ capsules each consisting of (in random order) the pair $u_{i,j} = \psi([0, \alpha_{i,j}]) = \alpha_{i,j}^r \bmod n$ and $v_{i,j} = \psi([1, \beta_{i,j}]) = y\beta_{i,j}^r \bmod n$.

Alice designates a random subset of the capsules and demands that they be opened.

Bob opens these capsules by presenting Alice with a decomposition of each component of the capsule.

For each remaining capsule $\{u_{i,j}, v_{i,j}\}$, Bob selects

$$\omega_{i,j} = \begin{cases} u_{i,j} & \text{if } c_i = 0; \\ v_{i,j} & \text{if } c_i = 1. \end{cases}$$

and shows that $\omega_{i,j}$ is of the same residue class as the associated $w_i$ by presenting Alice with a decomposition of the quotient $\omega_{i,j} w_i^{-1}$.

Alice tells Bob to which class each $w_i$ belongs.

Confidence: $1 - 1/2^N$

Security: $(1 - 1/2^{N'})^N > 1 - N/2^{N'}$

Figure 3.7: Interactive proof that $y \notin \mathbf{Z}_\mathbf{n}^\mathbf{r}$.

An important special case occurs when $r = 2$. Here Alice interactively proves to Bob that a given $y$ is not a quadratic residue modulo $n$. In this case,

in fact, there is a symmetry between multiplication and division, so the entire interactive proof can be completed without computing inverses. The interactive proof given here represents a significant simplification of the original interactive proof of quadratic non-residuosity given in [GMR85].

### 3.3.2 An Interactive Proof of Prime Consonance

The interactive proof that $y \notin \mathbf{Z}_{\mathbf{n}}^{\mathbf{r}}$ given in the previous section can also to used to show that a triple $(r, n, y)$ is prime consonant. This is true since, by theorem 2.18, when $r$ is prime and $y \in \mathbf{Z}_{\mathbf{n}}^{*}$, then $y \notin \mathbf{Z}_{\mathbf{n}}^{\mathbf{r}}$ is sufficient to imply that $(r, n, y)$ is prime consonant. The first two of these conditions are easily verifiable, so by entering into the interactive proof of section 3.3.1, Alice can convince Bob that a triple $(r, n, y)$ is prime consonant.

This interactive proof technique can be more efficient than the general interactive proof of consonance given in section 3.1.4. This is because, for each challenge $w$ presented to Alice by Bob, Alice need only decide the class of $w$ from among two possible residue classes ($\mathcal{RC}[0]$ and $\mathcal{RC}[1]$) instead of from among all $r$ possible residue classes.

### 3.3.3 Result-indistinguishable Residuosity

[GHY85] generalizes the result of [GMR85] in such a way that an observer, Carol, watching the protocol between Alice and Bob gains no information from the protocol as to whether Alice convinced Bob that a given $z$ was or was not a quadratic residue. The key addition to the protocol of [GMR85] is the inclusion of a third set of possibilities. Instead of choosing $w$ from among just the two sets $X = \mathcal{RC}[0]$ and $Y = \mathcal{RC}[1]$, Bob may select from an additional set $Z$. $Z$ consists of elements of the same class as $z$, and members of $Z$ can be randomly generated by multiplying $z$ by randomly chosen members of $\mathbf{Z}_{\mathbf{n}}^{\mathbf{2}}$.

By using three-component capsules, the protocol given in [GHY85] can be simplified tremendously. Bob simply prepares $N$ master capsules $C_i$, each consisting of one member of each of $X$, $Y$, and $Z$. For each $C_i$, Bob also prepares $N'$ additional scratch capsules of the same form. Alice designates some subset of the scratch capsules, and Bob opens these. Bob then matches the components of each scratch capsule with $C$. That is, on each scratch capsule, Bob links element of $X$ with the element of $X$ in capsule $C$, the element of $Y$ with

the element of $Y$ in capsule $C$, and the element of $Z$ with the element of $Z$ in capsule $C$. *Bob does not tell Alice which link is which.* Bob then demonstrates that each link is valid by decomposing the quotient of each pair of linked elements to show that each such quotient is a residue. Alice (now convinced that $C$ was generated as required) tells Bob which capsule component is of a class *different* from the other two — thus transmitting to Bob the class of $z$.

The chance of Alice being fooled into revealing excessive information to Bob is only 1 in $2^{N'}$. The chance of Alice fooling Bob in one iteration of this protocol is $1/2$, so by iterating the process $N$ times, Bob can obtain $1-1/2^N$ confidence that he has not been misled. Finally, a symmetry argument shows that Carol receives absolutely no information from watching this protocol that she could not have obtained on her own.

### 3.3.4 Graph Non-isomorphism

One example in which capsules are useful *without* the aid of residue classes is seen in a protocol for graph non-isomorphism given in [GMW86]. Their original protocol closely followed the non-residuosity protocol of [GMR85]. In [GMW86], a prover designates a graph $H$ given by the verifier as either a permutation of graph $G_1$ or of graph $G_2$ after being convinced that the verifier already holds such a permutation.

A simpler protocol is obtained by incorporating capsules in a manner similar that described in section 3.3.1. Residue classes are replaced by the equivalence classes induced by graph isomorphism, and class equivalence is demonstrated by exhibiting permutations. Thus, capsules — each consisting of a permutation of $G_1$ and a permutation of $G_2$ — are used to interactively prove that a given graph $H$ is a permutation of either $G_1$ or $G_2$.

### 3.3.5 Boolean Circuit Satisfiability

Recently (also in [GMW86]), Goldreich, Micali, and Wigderson gave a simple and elegant zero-knowledge interactive protocol (as defined in [GMR85]) to prove for any $k$ that a graph is $k$-colorable *without* revealing any information about a specific coloring (assuming that the prover possesses a $k$-coloring of the graph). Because $k$-colorability is $\mathcal{NP}$-complete, this implies that *any* positive

instance of a problem in $\mathcal{NP}$ for which a prover holds a certificate (e.g. a satisfying assignment for a Boolean formula) can be reduced to graph colorability and shown, in a zero-knowledge fashion, to be a positive instance. The only assumption made is the existence of a probabilistic cryptosystem, and this is implied by the existence of a one-way permutation (see [GoMi84] and[Yao82b]).

In this section, we shall examine an alternate approach which gives the same result by a very different method. The method uses capsules to give a zero-knowledge protocol to interactively prove that a given Boolean formula (or arbitrary Boolean circuit with in-degree 2) has a satisfying assignment. Brassard and Crepeau in [BrCr86] independently of both this work and [GMW86] have achieved the same result, and Chaum in [Chau86a] has also independently achieved a very similar result.

The major advantage of this method over the original is efficiency. When a Boolean formula or circuit is reduced to a colorability graph, the number of vertices and edges in the resulting graph is linear in the size of the Boolean formula. Each stage of the interactive proof protocol of Goldreich, Micali, and Wigderson, however, requires a new encryption of the entire graph; and for any fixed confidence level desired, their protocol requires a number of stages which is linear in the number of edges in the graph. Thus, the number of probabilistic encryptions required by this protocol grows quadratically with the size of the graph (or circuit). Because of the local nature of the method presented below, re-encryption is not necessary, and the number of probabilistic encryptions required grows only linearly with the size of the circuit (or graph).

The major disadvantage of this method compared to the original method is that the new procedure requires a (seemingly) stronger cryptographic assumption. Although both methods require a probabilistic encryption function such as the residue class based probabilistic encryption of [GoMi84], the method given here requires a probabilistic encryption function for which two encrypted values can be proven, in a zero-knowledge manner, to be encryptions of the same value. Although this property *is* easily achieved by the residue class based probabilistic encryption (Lemma 2.10), it is not at all obvious that every probabilistic encryption function has this property. However, by observing that the problem of inverting a probabilistic encryption function is itself in $\mathcal{NP}$, the original Goldreich, Micali, and Wigderson result can be applied to show that the cryptographic assumption required here is, in fact, no stronger than the

assumption of the existence of an arbitrary probabilistic cryptosystem.

**The Satisfiability Scheme**

The basic idea of the scheme is again deceptively simple. If Alice wants to prove to Bob that a given formula is satisfiable (and Alice has a satisfying assignment), Alice begins by choosing an $n$ which is the product of two large primes and providing Bob with $n$ and a $y$ (with Jacobi symbol 1) which is not a quadratic residue modulo $n$. This is merely the establishment of a Goldwasser–Micali probabilistic encryption function. The fact that $y$ has Jacobi symbol 1 (with respect to $n$) can be verified by Bob without Alice's assistance. Alice can convince Bob that $y \notin \mathbf{Z}_n^2$ by engaging in the non-residuosity protocol of section 3.3.1.

Alice then draws a circuit to compute the Boolean function (in the obvious way), selects a satisfying assignment and sends Bob an encryption of this assignment. For each variable, Alice sends Bob a random member of $\mathcal{RC}[0]$ if that variable is False/0/Off and a random member of $\mathcal{RC}[1]$ if that variable is True/1/On. Alice then encrypts the output of each gate of the circuit in the same manner and sends Bob these encrypted values as well.

For each gate in the circuit, Alice then interactively proves to Bob that the gate computes the required function. The computation of an AND gate will be shown here, and other Boolean functions should become apparent.

To prove that a given gate computes an AND on its inputs, a full truth table for AND is used. There are, of course, four possibilities: either both inputs and the output are 0; the first input is 0, the second is 1, and the output is 0; the first input is 1, the second is 0, and the output is 0; or both inputs and the output are 1. Four-component capsules are prepared such that each of these four input/output cases is represented by one of the four components of each capsule. Each case is represented by an *ordered* triple containing random members of the residue classes corresponding to these values. A capsule for AND would be a four-element (unordered) set consisting of four (ordered) triples whose elements are members of residue classes $(\mathcal{RC}[0], \mathcal{RC}[0], \mathcal{RC}[0])$, $(\mathcal{RC}[0], \mathcal{RC}[1], \mathcal{RC}[0])$, $(\mathcal{RC}[1], \mathcal{RC}[0], \mathcal{RC}[0])$, and $(\mathcal{RC}[1], \mathcal{RC}[1], \mathcal{RC}[1])$.

Let $G$ be an arbitrary gate with inputs marked with $\alpha$ and $\beta$ and an output marked with $\gamma$. Each of $\alpha$, $\beta$, and $\gamma$ is a random member of $\mathcal{RC}[c]$ if and only if the actual value of the input/output is $c$. Let $f$ be the function computed

by gate $G$. To prove that a gate $G = (\alpha, \beta, \gamma)$ computes the claimed function $f$, Alice must show that $f([\![\alpha]\!], [\![\beta]\!]) = [\![\gamma]\!]$. To accomplish this, Alice prepares many capsules of the form described for an AND gate above and Bob selects an arbitrary subset to be opened. Alice opens these capsules by giving Bob a decomposition of each element of each triple of each component of each capsule. Alice then proves that each unopened capsule matches $G$ by selecting one of its four components (call it $(\alpha', \beta', \gamma')$) and presenting Bob with a square root of each of the three quotients $\alpha'\alpha^{-1}$, $\beta'\beta^{-1}$ and $\gamma'\gamma^{-1}$.

Finally, Alice interactively proves that the output of the circuit is 1 by proving that this value is a non-residue as in section 3.3.1.

**Remark** Some gates may be computed without the need for an interactive proof. For example, an encrypted value may be complemented simply by multiplying it by $y$, and the XOR of two or more encrypted values is represented by their product.

If it were also possible to compute AND gates or OR gates without interactive proofs, then the satisfiability of any given circuit could be demonstrated with a single interactive proof that the value of the circuit output is 1. Unfortunately, no encryption homomorphism is known which allows the direct computation of an expressively complete set of boolean functions.

## 3.4 Basic Elections

We are now ready to make our second pass at secret-ballot elections. Most of the problems raised in the elementary election scenario of section 2.9 are addressed here. The centralized government does, however, remain. As an additional convenience, we assume here the existence of a universally trusted source of random bits — a so-called *beacon* (see [Rabi83a]). Although it will not be necessary to use a beacon, it is a simpler mechanism than that which will ultimately be given. A beacon may be produced as the outcome of some natural event which defies prediction such as the low order bit of a geiger-counter sampling in a radio-active environment, a measurement of sun-spot activity, or a seismographic reading.

In this scheme, $\mathcal{V}$ is the set of eligible voters, and $r$ is a fixed prime chosen such that $r > |\mathcal{V}|$.

The improvements over the elementary election scenario exhibited here are mainly due to the incorporation of interactive proofs to give confidence that, at various stages, agents are performing as required. In particular, votes are prepared by forming a capsule (or "ballot") consisting of a "no vote" and a "yes vote".

In this schema, an integer $w$ is said to be a no vote if $w \in \mathcal{RC}[0]$ and a yes vote if $w \in \mathcal{RC}[1]$. A capsule can be interactively proven to consist of a no vote and a yes vote by an interactive proof method which is nearly identical to the interactive proof that $[\![w]\!] \in \{0,1\}$ from section 3.3.1. Recall that in that interactive proof method, an integer $w$ is chosen uniformly from either $\mathcal{RC}[0]$ or $\mathcal{RC}[1]$ and interactively proven to be of that form by relating $w$ to capsules each of which contains one element from each of $\mathcal{RC}[0]$ and $\mathcal{RC}[1]$.

We could simply use the same interactive proof method and allow such a $w$ to then be used as the vote, but we wish to allow the interactive proof to be performed before the voter is forced to decide which way to vote. To do this easily, we allow each voter to prepare a "master" ballot (capsule) consisting of a no vote and a yes vote and many similar "scratch" ballots to be used in the interactive proof. A random subset of these capsules are demanded to be "opened" (decomposed), and each remaining scratch ballot is related to the master ballot by demonstrating that their components are of the same residue classes (either matched directly or matched in reverse order).

After this interactive proof is complete, the voter may select one of the two votes from the master ballot as the vote to be cast in the election. Other participants should be convinced that the vote cast is legitimate (either a no vote or a yes vote) without discovering which is the case.

The basic election schema is given in figure 3.8.

The function *check* used in the schema is defined to be good exactly when agents' actions are superficially consistent with their protocols. In particular, $\mathit{check}_V(\ell)$ is defined to be good if and only if a voter successfully completes the interactive proof in phase 2 to show the legitimacy of its master ballot and then selects a vote from its master ballot as the vote to be cast in phase 3. We do not want to include in the tally votes cast by voters who have not demonstrated their legitimacy.

The *check* function is simple for all agents to compute and can be precisely defined. Its precise definition is, however, quite cumbersome and will not be

*Phase 1 executed by government:*

1. Release a randomly chosen pair $(n, y)$ such that $(r, n, y)$ is exact consonant. Voters may challenge the consonance of $(r, n, y)$ by engaging the government in the interactive proof of section 3.3.2 up to $|n|$ times each.

*Phase 2 steps executed by each voter $V_\ell \in \mathcal{V}$:*

2a. Let $\eta = \lceil |n| \log_2 |\mathcal{V}| \rceil$. For $0 \leq i \leq \eta$, randomly select $\alpha_i, \beta_i \in \mathbf{Z}_n^*$ and release as ballot $B_i$ the capsule consisting of the two numbers $u_i = \psi([0, \alpha_i]) = \alpha_i^r \bmod n$ and $v_i = \psi([1, \beta_i]) = \beta_i^r \bmod n$ in random order.

2b. Use beacon to generate random bits $b_i for 1 \leq i \leq \eta$.

2c. For all $i$ such that $b_i = 0$, reveal $\alpha_i$ and $\beta_i$. For all $i$ such that $b_i = 1$, reveal $\alpha_i \cdot \alpha_0^{-1}$ and $\beta_i \cdot \beta_0^{-1} \bmod n$.

*Phase 3 executed by each voter $V_\ell \in \mathcal{V}$:*

3. Select one element of its $B_0$ as its actual vote $w_\ell$. To vote "yes", select $w_\ell = y\beta_0^r$. To vote "no", select $w_\ell = \alpha_0^r$.

*Phase 4 executed by government:*

4. Compute $\mathcal{G} = \{\ell : check_V(\ell) = \text{good}\}$ and $W = \prod_{\ell \in \mathcal{G}} w_\ell \bmod n$.

   Reveal $t = [\![W]\!]$ and use the interactive proof method of section 3.1.2 to show that $t \in \mathcal{RC}[W]$.

The value of $t$ represents the number of yes votes cast in the election.

Figure 3.8: The schema for elections with a government.

given here. A complete specification of the more general *check* function used in the full verifiable secret-ballot election schema will be given along with the full schema in chapter 5.

The advantages achieved by this election schema over the schema outlined in section 2.9 are mainly in the area of verifiability. The interactive proofs ensure the legitimacy of the government's encryption function and the votes cast and, ultimately, the correctness of the tally. Even collusion between dishonest voters and a corrupt government is not sufficient to convince honest voters of an incorrect tally.

There is also some measure of privacy since the vote cast by each voter is indecipherable by other voters. Thus, if the government can be trusted to maintain the privacy of votes, then the schema given here is sufficient to enable the conducting of verifiable secret-ballot elections.

The main difficulty left to be addressed is the government's ability to see how each voter voted. This alone gives a government extraordinary power to compromise privacy, to apply coercion to voters, or to halt elections which would yield an "undesirable" tally.

The methods of chapter 4 will show how to divide an entity such as a government into many constituent components such that no set of fewer than a pre-determined number of the components can obtain any useful information about a given process. Chapter 5 will show how this method can be applied to elections in order to prevent deciphering of individual votes.

# Chapter 4

# Secret Sharing Homomorphisms

In 1979, Blakley and Shamir independently proposed schemes by which a secret can be divided into many shares which can be distributed to mutually suspicious agents. If at least some pre-determined number of these agents pool their shares, they can reconstruct the secret. If, however, fewer than this number of agents pool their shares, they obtain no information whatsoever about the secret.

This chapter describes a homomorphism property attained by these and several other secret sharing schemes which allows multiple secrets to be combined by direct computation on shares. This property reduces the need for trust among agents and allows secret sharing to be applied to many new problems. One application described here gives a method of verifiable secret sharing which is much simpler and more efficient than previous schemes. A second application gives the final piece which allows the construction of a fault-tolerant method of holding verifiable secret-ballot elections.

## 4.1 Threshold Schemes

In 1979, Shamir in [Sham79] defined the notion of a *threshold* scheme (often called a *secret sharing* scheme).

Shamir defines a $(k, n)$ *threshold scheme* to be a division of a secret $D$ into $n$ pieces $D_1, \ldots, D_n$ in such a way that:

(1) knowledge of any $k$ or more $D_i$ pieces makes $D$ easily computable;

(2) knowledge of any $k-1$ or fewer $D_i$ pieces leaves $D$ completely undetermined (in the sense that all its possible values are equally likely).

The pieces of a secret are often called *shares.*

Since 1979, several different threshold schemes have been proposed (see [Blak79], [AsBl80], and [Koth84] for some examples), and they have proved to be useful in a wide variety of interactive protocols. The threshold scheme originally proposed by Shamir in [Sham79] is the best known and is very simple and elegant. It is based on polynomial interpolation and evaluation, and for completeness it will be sketched here.

### Shamir's Threshold Scheme

It is well known that given any field $\mathbf{F}$, any set of $k$ distinct values

$$x_1, x_2, \ldots, x_k \in \mathbf{F},$$

and $k$ additional values

$$y_1, y_2, \ldots, y_k \in \mathbf{F},$$

there exists a unique polynomial $P$ of degree at most $k-1$ with coefficients in $\mathbf{F}$ such that for all $i$, $P(x_i) = y_i$. This unique polynomial $P$ is said to be the *interpolation* of the points $(x_i, y_i)$.

Shamir's threshold scheme fixes $\mathbf{F}$ to be a finite field and enables the sharing of a secret $s \in \mathbf{F}$. To accomplish this, a polynomial of degree up to $k-1$ with constant term $s$ is randomly selected. This can be done by randomly selecting $k-1$ coefficients $c_i \in \mathbf{F}$ and forming

$$P(x) = c_{k-1}x^{k-1} + \cdots + c_1 x + s.$$

The $i^{\text{th}}$ share of $s$ is then computed as $t_i = P(i)$.

It is not hard to see that (as long as the number of shares is less than $|\mathbf{F}|$) the following conditions hold: (1) any $k$ shares are sufficient to interpolate the polynomial $P$ and thereby construct the secret $s$; and (2) $k-1$ or fewer shares leave $s$ completely undetermined since for any set of $k-1$ shares and any possible secret $s' \in \mathbf{F}$, there exists exactly one polynomial $P'$ which interpolates the $k-1$ known shares together with the point $(0, s')$ and which has degree at most $k-1$.

**Definition** For convenience, we use the notation $\|\langle t_1, t_2, \ldots, t_n\rangle\|$ to denote the secret defined by the shares $t_1, t_2, \ldots, t_n$ in Shamir's scheme. That is, $\|\langle t_1, t_2, \ldots, t_n\rangle\|$ is the value at 0 of the minimum degree polynomial $P$ passing through the points $(i, t_i)$.

We may now begin to formally define a threshold scheme.

**Definition** We say that $G : A \to B$ is a *random function* if for each $a \in A$, $G(a)$ is a random variable with a given probability distribution over $B$.

**Definition** Let $a \in A$ and $b \in B$ be elements of $A$ and $B$, respectively. We write $\text{Prob}(a \overset{G}{\mapsto} b)$ (read "the probability that $a$ yields $b$ under $G$") to denote $\text{Prob}(G(a) = b)$.

Let $S$ be the domain of possible secrets, and let $T$ be the share domain. The following definition asserts that given some partial set of shares, a given secret yields this set with a probability equal to that with which the secret yields some set of shares containing the partial set.

**Definition** Given a set $I = \{i_1, i_2, \ldots, i_\ell\} \subseteq \{1, 2, \ldots, n\}$ together with values $\tau_{i_1}, \tau_{i_2}, \ldots, \tau_{i_\ell}$ such that the value $\tau_{i_j}$ represents the $i_j^{\text{th}}$ value of an $n$-tuple in $T^n$, we define

$$\text{Prob}(a \overset{G}{\mapsto} \tau_{i_1}, \tau_{i_2}, \ldots, \tau_{i_\ell}) = \sum \text{Prob}(a \overset{G}{\mapsto} (t_1, t_2, \ldots, t_n))$$

where the sum is taken over all $n$-tuples in $T^n$ such that for all $i \in I$, $\tau_i = t_i$.

We are now ready to define a $(k, n)$ threshold scheme.

**Definition** A $(k, n)$ threshold scheme consists of a random function $G : S \to T^n$ and a collection of functions $F_I : T^k \to S$ defined for each $I \subseteq \{1, 2, \ldots, n\}$ with $|I| = k$.

These functions satisfy the following three properties:

(1) The functions $G$ and $F_I$ for all $I$ are computable in polynomial time.

(2) Let $(t_1, t_2, \ldots, t_n)$ be a value of $G(s)$ for some $s \in S$. Then for all $I = \{i_1, i_2, \ldots, i_k\} \subseteq \{1, 2, \ldots, n\}$ with $|I| = k$,

$$F_I(t_{i_1}, t_{i_2}, \ldots, t_{i_k}) = s.$$

(3) Let $t_{i_1}, t_{i_2}, \ldots, t_{i_{k-1}}$ be a collection of $k-1$ shares. Then for all $s, s' \in S$,

$$\text{Prob}(s \overset{G}{\mapsto} t_{i_1}, t_{i_2}, \ldots, t_{i_{k-1}}) = \text{Prob}(s' \overset{G}{\mapsto} t_{i_1}, t_{i_2}, \ldots, t_{i_{k-1}}).$$

The first condition implies that secrets can be efficiently divided into shares and reconstructed from the shares. The second implies that *any* subset consisting of $k$ of the $n$ shares determines the same secret as does any other subset. Finally, the third condition implies that no set of fewer than $k$ shares gives any information whatsoever about the secret—even if partial *a priori* information about the secret exists.

## 4.2 The Homomorphism Property

**Definition** Let $\oplus$ and $\otimes$ be arbitrary binary functions on elements of the secret domain $S$ and of the share domain $T$, respectively. We say that a $(k, n)$ threshold scheme has the $(\oplus, \otimes)$-*homomorphism* property (or is $(\oplus, \otimes)$-*homomorphic*) if for all $I$, whenever

$$s = F_I(t_{i_1}, t_{i_2}, \ldots, t_{i_k})$$

and

$$s' = F_I(t'_{i_1}, t'_{i_2}, \ldots, t'_{i_k}),$$

then

$$s \oplus s' = F_I(t_{i_1} \otimes t'_{i_1}, \ldots, t_{i_k} \otimes t'_{i_k}).$$

This property implies that the compositions of the shares of the secrets are shares of the composition of the secrets.

The following diagram shows how a set of $m$ secrets $s_1, s_2, \ldots, s_m$ can be divided into shares such that $t_{i,j}$ represents the $i^{\text{th}}$ share of the $j^{\text{th}}$ secret $s_j$.

$$\begin{array}{ccccccccc}
s & = & s_1 & \oplus & s_2 & \oplus & \cdots & \oplus & s_m \\
\hline
t_1 & = & t_{1,1} & \otimes & t_{1,2} & \otimes & \cdots & \otimes & t_{1,m} \\
t_2 & = & t_{2,1} & \otimes & t_{2,2} & \otimes & \cdots & \otimes & t_{2,m} \\
\vdots & & \vdots & & \vdots & & & & \vdots \\
t_n & = & t_{n,1} & \otimes & t_{n,2} & \otimes & \cdots & \otimes & t_{n,m}
\end{array}$$

Each $t_i$ is computed as the composition (under $\otimes$) of all of the $i^{\text{th}}$ shares. The value $s$ is computed as the composition (under $\oplus$) of the secrets $s_j$, and this value is also computable as the secret defined by the "composite shares" $t_i$.

**Definition** The value $s$ formed as the composition (under $\oplus$) of the secrets $s_1, s_2, \ldots, s_m$ is called the *composite secret* and the secrets $s_1, s_2, \ldots, s_m$ are called *constituent secrets.* The shares $t_{i,j}$ of the constituent secrets are called *constituent shares*, and the values $t_i$ formed as the compositions (under $\otimes$) of the constituent shares are called *composite shares.*

The homomorphism property can now be restated as *"the composite shares are shares of the composite secret."*

**Lemma 4.1** *Shamir's threshold scheme is* $(+,+)$*-homomorphic where the addition is defined over the integers modulo* $p$*.*

*Proof:*

Let $P_1(x)$ and $P_2(x)$ be two polynomials each of degree at most $k-1$ for some positive integer $k$. Recall that with Shamir's scheme, these polynomials define the secrets $P_1(0)$ and $P_2(0)$, respectively, and the $i^{\text{th}}$ shares of these secrets are given by $P_1(i)$ and $P_2(i)$, respectively.

Now, consider the polynomial $P(x) = P_1(x) + P_2(x)$. Since both $P_1(x)$ and $P_2(x)$ are of degree at most $k-1$, their sum $P(x)$ is also of degree at most $k-1$. Also, for all $i$, $P(i) = P_1(i) + P_2(i)$. By the interpolation theorem, for any subset of $k$ of the points $(i, P(i))$, there is a unique polynomial of degree at most $k-1$ which interpolates these points. Since $P(x)$ is of the proper degree and passes through these points, every set of $k$ points $(i, P(i))$ interpolates to the polynomial $P(x)$. This polynomial determines the secret $P(0)$ which is equal to $P_1(0) + P_2(0)$. Thus, Shamir's scheme is $(+,+)$-homomorphic. ∎

## 4.3 Composite Security

One use for the homomorphism property of section 4.2 is to allow for the computation of a composite secret without ever revealing the constituent secrets. This may be desirable in a number of circumstances in which computation on secret data is desired. [RAD78], [Yao82a], [Feig85], and [GMW87] discuss problems of this sort, and it will be seen that verifiable secret sharing and verifiable secret-ballot elections can be modeled in this way.

It seems, however, that the homomorphism property is not strong enough for these applications. We want it to also be the case that up to $k-1$ complete sets of constituent shares together with *all* of the composite shares (and

therefore the composite secret) give no more information about the constituent secrets than does the composite secret alone. Thus, if a $(\oplus, \otimes)$-homomorphic threshold scheme is used to generate a composite secret from the shares of constituent secrets, then no set of up to $k-1$ dishonest shareholders could gain any information at all about the constituent secrets *other than* that which is given by the composite secret alone. Since it is assumed that the composite secret is to be given (and that it might give some partial information about the constituent secrets), this is the best that could be hoped for.

Without this property, it is conceivable that conspiring shareholders could gather the information released by honest shareholders (i.e. the composite shares) and use this information in order to construct at least partial information about the constituent secrets. If conspirators could accomplish this, then there is no point to composing the shares and releasing only a composite share, as the only purpose of this composition is to protect the constituent secrets.

Assume without loss of generality that the first $k-1$ shareholders wish to conspire in order to obtain additional information about the constituent secrets $s_1, s_2, \ldots, s_m$. The following diagram shows all of the values directly available to the conspiring shareholders.

| $s$ | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $t_1$ | $=$ | $t_{1,1}$ | $\otimes$ | $t_{1,2}$ | $\otimes$ | $\cdots$ | $\otimes$ | $t_{1,m}$ |
| $t_2$ | $=$ | $t_{2,1}$ | $\otimes$ | $t_{2,2}$ | $\otimes$ | $\cdots$ | $\otimes$ | $t_{2,m}$ |
| $\vdots$ | | $\vdots$ | | $\vdots$ | | | | $\vdots$ |
| $t_{k-1}$ | $=$ | $t_{(k-1),1}$ | $\otimes$ | $t_{(k-1),2}$ | $\otimes$ | $\cdots$ | $\otimes$ | $t_{(k-1),m}$ |
| $t_k$ | | | | | | | | |
| $t_{k+1}$ | | | | | | | | |
| $\vdots$ | | | | | | | | |
| $t_n$ | | | | | | | | |

We want it to be the case that these values do not assist the conspirators in gaining information about the values $s_1, s_2, \ldots, s_m$ beyond that which is given by $s$ alone.

**Definition** We say that a $(\oplus, \otimes)$-homomorphic threshold scheme is $(\oplus, \otimes)$-*composite* if for every table of the above form and for all values $s_j$, the proba-

bility that the $s_j$ represent the constituent secrets is the same as this probability when given only the value of the composite secret $s$.

The following theorem is somewhat surprising,

**Theorem 4.2** *If the secret domain $S$ and the share domain $T$ are finite and of the same cardinality, then every $(\oplus,\otimes)$-homomorphic $(k,n)$ threshold scheme is a $(\oplus,\otimes)$-composite $(k,n)$ threshold scheme.*

*Proof:*

Consider the following diagram of values available to the first $k-1$ shareholders.

$$
\begin{array}{c}
s \\
\hline
\begin{array}{lcccccccc}
t_1 & = & t_{1,1} & \otimes & t_{1,2} & \otimes & \cdots & \otimes & t_{1,m} \\
t_2 & = & t_{2,1} & \otimes & t_{2,2} & \otimes & \cdots & \otimes & t_{2,m} \\
\vdots & & \vdots & & \vdots & & & & \vdots \\
t_{k-1} & = & t_{(k-1),1} & \otimes & t_{(k-1),2} & \otimes & \cdots & \otimes & t_{(k-1),m} \\
t_k & & & & & & & & \\
t_{k+1} & & & & & & & & \\
\vdots & & & & & & & & \\
t_n & & & & & & & &
\end{array}
\end{array}
$$

By the definition of a threshold scheme, the $k-1$ sets of constituent shares $t_{i,j}$, for $1 \leq i < k$, give no information about the constituent secrets $s_j$ even though partial *a priori* information about the $s_j$ exists because the value of the composite secret $s$ is given.

Now, the constituent shares $t_{i,j}$ allow the computation of the composite shares $t_i$, for $1 \leq i < k$ since each $t_i$ is given by

$$t_i = t_{i,1} \otimes t_{i,2} \otimes \cdots \otimes t_{i,m}.$$

Thus, these composite secrets give no additional information about the constituent secrets $s_j$.

Finally, we observe that when the secret domain and the share domain are both finite and of the same cardinality, then the secret together with any set of $k-1$ shares completely determine the remaining shares. This is so because every secret is completely consistent with any set of $k-1$ shares. Thus, for each remaining share, there must be at least one value which would yield each possible secret. Therefore, there must be exactly one value of each remaining

share associated with each possible secret. Hence, the secret together with $k-1$ shares completely determine the remaining shares. This then implies that the remaining composite shares $t_i$, for $i \leq k \leq n$, are also computable by the first $k-1$ shareholders (when given the composite secret $s$). Hence, the composite shares also give no additional information about the constituent secrets $s_j$. ∎

**Remark** The condition that the secret domain $S$ and the share domain $T$ are of the same finite cardinality was not strictly required. In fact, almost any domain with a group structure would suffice. For simplicity of exposition (and to keep the notation under control), this generalization has not been incorporated into the theorem.

## 4.4 Some Examples

Lemma 4.1 shows that Shamir's $(k, n)$ threshold scheme is $(+, +)$-homomorphic over the integers modulo $p$ and since the secret domain and the share domain consist of the same finite set (the integers modulo $p$), theorem 4.2 implies that Shamir's scheme is $(+, +)$-composite.

Many other known threshold schemes are also $(+, +)$-homomorphic. The threshold schemes found in [Blak79], [AsBl80], and [Koth84], for example, are all $(+, +)$-homomorphic.

What if the desired composite secret is not the sum of the constituent secrets? Shamir's scheme is *not* $(\times, \times)$-composite. This is because the product of two non-constant polynomials is of higher degree than the factors.

By using a homomorphism between addition and discrete logarithms, for example, it *is* possible to transform Shamir's scheme into a $(\times, +)$-homomorphic $(k, n)$ threshold scheme. Thus, if the desired composite secret is the product of the constituent secrets, Shamir's scheme can still be used. This method can be summarized by the following statement. *The sums of the shares of the discrete logs of the secrets are shares of the discrete log of the product of the secrets.*

In general, discrete logarithms may be difficult to compute. However, if $p$ is small or of one of a variety of special forms, the problem is tractable (see [PoHe78], [Adle79], [COS86]). It should be emphasized that such special cases for $p$ do not in any way weaken the security of our schemes. The security is

*not* cryptographic, but rather is information theoretic. Therefore, there need be no assumptions about the difficulty of solving any special problems.

There is, however, a subtlety here, since discrete logarithms are only defined within multiplicative groups and the multiplicative subgroup $\mathbf{Z}_n^*$ has order $\varphi(n)$ which is *not* prime for prime $n > 4$. Polynomial interpolation, however, is only well-defined over a field which, if finite, must be of prime order.

To alleviate this difficulty, we let the secret domain $S$ be the set of $r^{\text{th}}$ roots of 1 modulo $n$ where $r$ is prime, $r \mid \varphi(n)$, and $r$ and $\varphi(n)/r$ are relatively prime. By theorem 2.20, this is sufficient to ensure that $S$ has exactly $r$ members, and since $r$ is prime, discrete logarithms over $S$ will yield values in $\mathbf{Z}_r$ which is a field.

## 4.5 Verifiable Secret Sharing

An important application of secret sharing homomorphisms provides a simple and efficient method for verifiable secret sharing. This problem was first described by Chor, Goldwasser, Micali, and Awerbuch in [CGMA85] and the application of secret sharing homomorphisms to this problem was developed as a result of an observation made by Oded Goldreich.

**Definition** We say that a set of $n$ shares $t_1, t_2, \ldots, t_n$ is *$k$-consistent* if every subset of $k$ of the $n$ shares defines the same secret.

The problem of verifiable secret sharing is to convince shareholders that their shares (collectively) are $k$-consistent. This task usually falls upon the holder of the initial secret $s$, and this agent is usually referred to as the *dealer*.

In Shamir's threshold scheme, the shares $t_1, t_2, \ldots, t_n$ are $k$-consistent if and only if the interpolation of the points $(1, t_1), (2, t_2), \ldots, (n, t_n)$ yields a polynomial of degree *at most* $d = k - 1$. It is useful to observe that if the sum of two polynomials is of degree at most $d$, then either both are of degree at most $d$ or both are of degree greater than $d$.

This suggests the following outline of an interactive proof that a polynomial $P$, given by its (encrypted) values at $n$ distinct points, is of degree at most $d$.

1. Encryptions of the values of the points that describe $P$ are released by the prover.

2. Encryptions of many additional random polynomials, again of degree at most $d$, are also released by the prover.

3. A random subset of the random polynomials is designated by the verifier(s).

4. The polynomials in the chosen subset are decrypted by the prover. They must all be of degree at most $d$.

5. Each remaining random polynomial is added to $P$. (Note that pointwise addition gives the same polynomial as the coefficientwise addition.) Each of these sum polynomials is decrypted by the prover. They must also all be of degree at most $d$.

Since the encryptions of shares are made public, the security is no longer information theoretic, but rather depends upon a cryptographic assumption. The encryption of the values of each point must be probabilistic to prevent guessing of values, and it must satisfy a homomorphism property so that an encryption of the sum of two values can be developed directly from the encryptions of the two values. These properties are satisfied by the encryption function $\mathcal{E}$ of section 2.7.

An efficient verifiable secret sharing scheme can thus be given as follows.

Let $r$ be a fixed prime, and suppose the secret value $s$ is to be drawn from $\mathbf{Z_r}$. Suppose further that each (future) shareholder has developed a public election encryption function $\mathcal{E}_i$ by releasing a pair $(n_i, y_i)$ such that $(r, n_i, y_i)$ is prime consonant, and has interactively proven that $(r, n_i, y_i)$ is prime consonant using the interactive proof method of section 3.3.2.

The dealer divides a secret $s$ into $n$ shares $t_1, t_2, \ldots, t_n$ using Shamir's secret sharing scheme. That is, coefficients $c_1, c_2, \ldots, c_{k-1}$ are randomly selected from $\mathbf{Z_r}$, $P(x) = c_{k-1}x^{k-1} + \cdots + c_2x^2 + c_1x + s$ is formed, and the shares $t_i = P(i)$ are computed. The $i^{\text{th}}$ share, $t_i$, is transmitted to the $i^{\text{th}}$ shareholder by releasing a randomly chosen member of $\mathcal{RC}[t_i]_{(r,n_i,y_i)}$. This is done by selecting a random $x_i \in \mathbf{Z}^*_{\mathbf{n_i}}$ and releasing $\mathcal{E}_i(t_i, x_i) = y_i^{t_i}x_i^r \bmod n_i$. Thus for each $i$, the pair $(i, \mathcal{E}_i(t_i, x_i))$ is made public.

To convince a participant that the (encrypted) points $(i, \mathcal{E}_i(t_i, x_i))$ describe a polynomial with degree no more than $d = k - 1$, the dealer can engage in the interactive proof of figure 4.1.

*Dealer:*

Select $m$ random polynomials $P'$, each of degree at most $d$. The values of these random polynomials at 0 (the secrets they describe) are also selected randomly. This is accomplished by selecting, for each random polynomial $P'$, $k$ coefficients $c_0, c_1, \ldots, c_{k-1}$ uniformly from $\mathbf{Z_r}$ and forming $P'(x) = c_{k-1}x^{k-1} + \cdots + c_1 x + c_0$. For each of these polynomials $P'$, select random values $x'_i \in \mathbf{Z}^*_{\mathbf{n_i}}$ and set $t'_i = P'(i)$. Reveal to the public the pairs $(i, \mathcal{E}_i(t'_i, x'_i))$.

*Verifier(s):*

Randomly select a subset $A$ of these random polynomials.

*Dealer:*

Each polynomial in $P' \in A$ is opened by revealing, for all $i$, the decomposition $[t'_i, x'_i]$. For each polynomial $P' \notin A$, the (pointwise) sum $P + P' \bmod r$ is opened by releasing, for all $i$, the pair $[t_i, x_i] \oplus [t'_i, x'_i]$ (as defined in section 2.6).

Figure 4.1: Interactive proof that $P(x)$ is of degree at most $d$.

By construction, each pair $[t'_i, x'_i]$ is a decomposition of the corresponding public value $\mathcal{E}_i(t'_i, x'_i)$. By releasing all $i$ decompositions (for each polynomial $P' \in A$), the dealer allows the polynomial $P'$ defined by the points $(i, t'_i)$ to be interpolated.

By lemma 2.8 and by the definition of the $\oplus$ operator, each of the pairs $[t_i, x_i] \oplus [t'_i, x'_i]$ is a decomposition of the publically computable product of encryptions given by $\mathcal{E}_i(t_i, x_i)\mathcal{E}_i(t'_i, x'_i)$. Recall that the first component of $[t_i, x_i] \oplus [t'_i, x'_i]$ is simply $t_i + t'_i \bmod r$. Thus, the first component of this decomposition of $\mathcal{E}_i(t_i, x_i)\mathcal{E}_i(t_i, x'_i)$ gives the sum $t_i + t'_i \bmod r = P(i) + P'(i) \bmod r$. Since Shamir's scheme is $(+,+)$-homomorphic (i.e., since the pointwise sum of

polynomials matches the coefficientwise sum), the dealer, by releasing each of the $n$ pointwise sums $P(i) + P'(i) \bmod r$ (for each polynomial $P' \notin A$), allows the polynomial $P + P' \bmod r$ to be interpolated.

Any verifier(s) can then confirm that all polynomials $P' \in A$ and all polynomials $P + P' \bmod r$ for $P' \notin A$ are of degree at most $d$. This should be sufficient to convince the verifier(s) with confidence $1 - 1/2^m$ that $P$, too, is of degree at most $d$, as desired.

If there is more than one verifier (in particular if each shareholder is to be convinced of the consistency of the shares), then each verifier can enter into such an interactive proof with the dealer, or the verifiers may all take part in the interactive proof by jointly selecting the subset of random polynomials to be opened by the dealer.

It is not hard to see that a set of random polynomials of degree at most $d$ together with a set of sums of $P$ with other random polynomials of degree at most $d$ gives no useful information about $P$ beyond the degree bound. This is, in fact, a special case of theorem 4.2.

Thus, by combining Shamir's secret sharing scheme with the results of this chapter and chapter 2 (and using the interactive proof methods given in chapter 3), a simple and efficient verifiable secret sharing scheme can be produced.

# Chapter 5

# Secret-Ballot Elections

We are now ready to combine the tools described in the preceding chapters and produce a schema for verifiable secret-ballot elections. In the schema described in this chapter, voters cast their votes in encrypted form, and a "government" or a set of constituent "tellers" releases a tally and a proof of its correctness which can be verified by all. It is proven that if a set of conspiring voters can, in polynomial time, obtain more than a small advantage at compromising the privacy of the honest voters, then the prime residuosity assumption of chapter 2 is false.

## 5.1 Overview of Secret-Ballot Elections

The major shortcomings of the basic election schema described in section 3.4 can be summarized by the fact that the central government is too powerful. The government can tell how individual voters cast their votes and can, if it wishes, halt an election any time it likes. To alleviate these problems, the basic election schema can be embedded within a $(+,+)$-composite $(J,K)$ threshold scheme (in particular, in Shamir's scheme) as suggested by the outline below.

Instead of a single government, $K$ sub-governments (or *tellers*) each hold a "sub-election". Each voter chooses either 0 or 1 as a secret value (0 indicating a no vote, 1 indicating a yes vote) and distributes one share of the secret vote to each of the $K$ tellers. The tally of the election is given by the sum of the voters' secrets.

Since the $(J,K)$ threshold scheme has the $(+,+)$-homomorphism property,

the sum of "vote-shares" is itself a share of the sum (tally) of the votes. Thus, once $J$ or more tellers release their "tally-shares", the overall election tally can be determined. Furthermore, since the secret domain and the share domain consist of the same finite set, the conditions of theorem 4.2 are satisfied, and fewer than $J$ conspiring tellers are unable to determine any individual voter's secret vote.

In order to maintain privacy of the vote-shares (and hence the votes themselves) while allowing the accuracy of the tally-shares (the sums of the vote-shares) to be verified, the single-government verifiable selection schema of section 3.4 is used. Each teller selects an election encryption function $\mathcal{E}_i$, and each voter transmits the $i^{\text{th}}$ share of its vote by revealing an encryption of this share under the $i^{\text{th}}$ teller's encryption function $\mathcal{E}_i$. Each teller then computes the sum of the vote-shares it receives and (while keeping the actual vote-shares secret) interactively proves that the tally-share it releases represents the sum of these vote-shares by decrypting the product of the encrypted vote-shares. Thus, even though the vote-shares cast by each voter are no longer constrained to being 0 or 1, each teller can produce a verifiable sum of these by shares as in the single-government election schema of section 3.4.

The interactive proof techniques used in section 4.5 can be generalized slightly to allow verification of the vote-shares. Here, each voter participates in an interactive proof to demonstrate to all participants that the vote-shares it distributes are *legitimate* in the sense that every set of $J$ of its vote-shares derives the same secret vote (the set of vote-shares is $J$-consistent) and that this vote is either a 0 or a 1.

Thus, as long as at least $J$ of the $K$ designated tellers participate through to conclusion, an election can be conducted such that each participant has very high confidence in the accuracy of the resulting tally and no set of fewer than $J$ tellers, together with any number of conspiring voters, can gain as much as an inverse polynomial advantage at distinguishing between possible votes of honest voters without breaking the underlying cryptosystem and thereby violating the prime residuosity assumption.

A "normal" threshold which may be used for an election might set $J$ to be about 90%-95% of $K$. In this scenario, a small fraction (5%-10%) of tellers could fail without causing the election to fail, and a large fraction (90%-95%) of the tellers would have to collude to resolve individual votes.

## 5.2 Related Work

Various cryptographic schemas have been proposed for boardroom voting in which participants pass encrypted messages from one to another while performing encryption and decryption operations until a certain point is reached at which all are confident of the outcome of the vote (see [DLM82], [Merr83], and [Yao82a]). These all share the problems that the active participants must be known in advance and if one participant stops following its protocol during the election, the election cannot continue. Thus these boardroom schemas are not well suited for large-scale elections.

Chaum ([Chau81]) proposes the use of a trusted "mix" (similar to a "government") to scramble pairs of votes and digital pseudonyms. The votes are publicly revealed, but the identity of the corresponding voters is protected by the mix. This mix schema seems to have properties very similar to those of the single-government election schema presented in section 3.4, but the approach is very different. Instead of hiding voters, the schemas presented in this thesis hide the actual values of the votes. Chaum also suggests the use of multiple mixes cascaded to improve the privacy of individual voters. The properties obtained with the cascaded mix schema appear to be similar to those proven for the multiple teller schema presented in this chapter *when no faulty tellers are allowed.* If one mix fails, however, the election collapses whereas the failure of any predetermined number of tellers can be tolerated in the secret-ballot election schema presented here.

Very recently, Chaum ([Chau86b]) has given another method of holding verifiable secret-ballot elections which obviates the need for a mix. The work is similar to boardroom voting in that failure of a single voter causes an election to fail; however, Chaum's method ensures that such failures can be traced. This allows an election to be restarted without the faulty voter. This approach, however, still does not seem practical for large-scale elections since the election protocols might have to be repeated once for each faulty or malicious voter in the system until an election with no improper voters can be completed.

Also very recently, Goldreich, Micali, and Wigderson ([GMW86]) found a method to show that *all* $\mathcal{NP}$ predicates can be proven in a "zero-knowledge" fashion. With respect to secret-ballot elections, their result can be used to give a method of verifiable secret-ballot elections, but there are limitations. Firstly,

their result can only tolerate failure of half of the participants, whereas any number of dishonest or faulty tellers can be tolerated by the schema presented here. Secondly, the methods of Goldreich, Micali, and Wigderson are, although polynomially bounded, far beyond practicality, while the given schema is shown to be within the bounds of practicality with current technology.

The first version of this work was given by Cohen (Benaloh) and Fischer in [CoFi85]. Various improvements and extensions were given by Cohen (Benaloh) in [Cohe86] and by Benaloh and Yung in [BeYu86]. The verifiable secret-ballot election schema to be presented in this chapter incorporates most of the ideas presented in these previous versions and adds new methods which both improve the results and simplify the presentation.

## 5.3 Election Definitions

We now give a formal model of the verifiable secret-ballot election problem and the properties that we want a solution to have.

### 5.3.1 Verifiable Elections

A *J-threshold K-teller L-voter election system* (or simply a $(J, K, L)$ *election system*) $\mathcal{E}$ is a synchronous system of communicating processes. The processes may be thought of as probabilistic Turing machines extended with operations for communication. The program run by such a process is called a *protocol*. $K$ of the processes, $T_1, \ldots, T_K$ are designated as tellers, and $L$ of the processes $V_1, \ldots, V_L$ are designated as (potential) *voters*. We denote the set of teller processes by $\mathcal{T}$ and the set of voter processes by $\mathcal{V}$. $\mathcal{T}$ and $\mathcal{V}$ are fixed in advance, and each process knows the designation of every process.

Communication is via *bulletin boards* which can be thought of as restricted shared memories. Each process controls one bulletin board, which it is said to *own*. The correspondence between bulletin boards and processes is fixed in advance and known to all processes. Each bulletin board can be read by every process, but it can only be written by its owner, and then only by appending new messages, not by altering old ones. In practice, implementing such bulletin boards may be a problem unto itself. See [Fisc83] for a survey of the literature on this problem as well as [BenO83] and [Rabi83b] for some probabilistic

approaches.

In addition, there is a publicly-readable number $N$ called a *security parameter* which serves as the (only) input to each process. $N$ controls the likelihood that the election is correct and that privacy is maintained.

**Definition** A $(J, K, L)$ *election schema* $S$ consists of a collection of protocols for use by a $(J, K, L)$ election system and a function *check*. The function *check* returns either good or bad and depends on $N$ and the messages posted to the public bulletin boards. *check* must be computable in time polynomial in $N$.

$S$ prescribes a protocol $\pi_T$ for each teller process and two possible protocols for each voter: $\pi_{\mathsf{yes}}$ to be used to cast a "yes" vote and protocol $\pi_{\mathsf{no}}$ to be used to cast a "no" vote.

**Definition** An *election* under $S$ consists of a run of a $(J, K, L)$ election system $\mathcal{E}$ for which *check* returns good. Any process of $\mathcal{E}$ which follows (one of) its protocol(s) prescribed by $S$ is said to be *proper*; otherwise it is *improper*. We say that a voter *casts a valid "yes" vote* (respectively *"no" vote*) if the messages it posts are consistent with the protocol $\pi_{\mathsf{yes}}$ (respectively $\pi_{\mathsf{no}}$). We say it *votes properly* if it casts a valid "yes" or "no" vote; otherwise it *votes improperly.* Note that a proper voter by definition always votes properly, but an improper voter may or may not vote properly, and if it votes improperly, that fact may or may not be detectable by others.

**Definition** The *tally* of an election is the pair $(t_{\mathsf{yes}}, t_{\mathsf{no}})$ where $t_{\mathsf{yes}}$ and $t_{\mathsf{no}}$ are the number of voters who cast valid "yes" and "no" votes, respectively. As part of its protocol, each teller $T_k$ releases a value $\tau_k$. For some pre-determined function $\Gamma$, if $\Gamma(\tau_1, \ldots, \tau_K) = (t_{\mathsf{yes}}, t_{\mathsf{no}})$, then the tally of the election is said to be correct.

Let $\delta$ be a function of $N$. The schema $S$ is said to be *verifiable with confidence* $1 - \delta$ if, for any election system $\mathcal{E}$, *check* satisfies the following properties for random runs of $\mathcal{E}$ using security parameter $N$:

1. If at least $J$ tellers are proper in $\mathcal{E}$, then, with probability at least $1-\delta(N)$, *check* returns good and the tally of the election is correct.

2. The joint probability that *check* returns good and the election tally is not correct is at most $\delta(N)$ .

$S$ is said to be *verifiable* if for every inverse polynomial function $\delta(N) = 1/P(N)$, there exists an integer $N_0$ such that $S$ is verifiable with confidence $1 - \delta(N)$ whenever $N \geq N_0$

### 5.3.2 Public Voting

A simple example of a verifiable election schema is one in which each voter publicly posts a single "yes" or "no" vote. Each teller $T_k$ then counts the "yes" and "no" votes and announces the totals as $\tau_k$. If at least $J$ of the $K$ tellers post totals $\tau_k$ which match the number of yes and no votes posted, then $\Gamma(\tau_1, \ldots, \tau_K)$ is defined to be this value $\tau_k$. Otherwise, $\Gamma$ is undefined.

*check* returns good if and only if the totals of the valid votes are the same as those announced by at least $J$ of the tellers. Thus, by computing the function *check*, any participant can verify the accuracy of the announced tally.

### 5.3.3 Privacy

The trivial example above shows that a verifiable election schema is not very interesting without the incorporation of some notion of privacy. Preserving privacy in an election does not always imply the inability of one voter to determine another's vote. For example, in the case of a unanimous mandate, every voter knows every other voter's vote. More generally, any coalition of voters can determine the sub-tally of the votes cast by voters outside of the coalition simply by subtracting their own votes from the released totals. We say that privacy is maintained if any conspiracy of voters and tellers which does not include at least $J$ tellers has at most a small advantage at distinguishing between any two vote assignments that have the same sub-tally on a set of proper voters. In particular, if there exist two proper voters of which one casts a yes vote and one casts a no vote, then no conspiracy of fewer than $J$ tellers and any number of remaining voters can attain more than a small advantage at deciding which of these two votes is the yes vote.

In order to account for the possibility of a collection of improper voters acting in concert, we augment our model to permit private communication channels between improper processes. However, we do not want to assume that private channels are always available, so we do not permit their use in the

protocols prescribed by a verifiable election schema. In other words, the adversaries can communicate secretly among themselves but the proper processes cannot.

To formalize the privacy requirement, let $C_{\mathcal{T}} \subset \mathcal{T}$ (the conspiring tellers) such that $|C_{\mathcal{T}}| < J$, and let $C_{\mathcal{V}} \subseteq \mathcal{V}$ (the conspiring voters). Let $C = C_{\mathcal{T}} \cup C_{\mathcal{V}}$ be the set of conspiring processes, and let $c_0 \in C$. We define a $(c_0, C)$-*conspiracy* $\mathcal{C}$ to consist of an assignment of protocols to processes in $C$ (possibly with private communication among themselves) such that $c_0$ produces an output in $\{0,1\}$ which we denote "output($\mathcal{C}$)". We require the running time of each process in $C$ to be polynomial in $N$.

We now define what it means for a conspiracy $\mathcal{C}$ to compromise privacy of an $(J,K,L)$ election schema $\mathcal{S}$. Let $H = \mathcal{V} - C_{\mathcal{V}}$ (the honest voters), and let $h_0$ and $h_1$ be assignments of votes to voters in $H$ such that the sub-tally of votes in $h_0$ is the same as that in $h_1$. For each $i \in \{0,1\}$, define an election system $\mathcal{E}_i$ as follows. Every process in $C$ runs the protocol assigned to it by $\mathcal{C}$, every voter process $V_\ell \in H$ runs protocol $\pi_{h_i(\ell)}$, and every teller process $T_k$ not in $C_{\mathcal{T}}$ runs its proper protocol $\pi_{\mathcal{T}}$. For a fixed security parameter $N$, let $p_i$ be the probability that output($\mathcal{C}$) $= 1$ on a random run of $\mathcal{E}_i$, and let $\varepsilon$ be a real number. We say that $\mathcal{C}$ *distinguishes* $h_0$ *from* $h_1$ *with* $\varepsilon$ *advantage* if

$$|p_1 - p_0| > \varepsilon.$$

In other words, the conspiracy has an $\varepsilon$ advantage in determining whether the votes correspond to assignment $h_0$ or to $h_1$ in a given election with security parameter $N$.

**Definition** We say that $\mathcal{C}$ *compromises the privacy of* $(h_0, h_1)$ *in* $\mathcal{S}$ if, for some inverse polynomial function $\varepsilon = 1/P(N)$, $\mathcal{C}$ distinguishes $h_0$ from $h_1$ with $\varepsilon(N)$ advantage on infinitely many values of $N$. Finally, we say that $\mathcal{S}$ is *secure* if for every $(c_0, C)$-conspiracy $\mathcal{C}$ and every pair of vote assignments $h_0, h_1$ to voters in $H$ that have the same sub-tally, $\mathcal{C}$ does not compromise the privacy of $(h_0, h_1)$ in $\mathcal{S}$.

The *election problem* is to find, for each triple $(J,K,L)$, a $(J,K,L)$ election schema that is verifiable and secure.

## 5.4 An Election Paradigm

The election paradigm upon which the schema presented here is based operates in four phases and is shown in Figure 5.1. The participants are a set of tellers $\mathcal{T} = \{T_1, T_2, \ldots, T_K\}$ and a set of voters $\mathcal{V} = \{V_1, V_2, \ldots, V_L\}$. Implicit in the paradigm is that each phase must be completed by all participants before the next begins. This is achieved by setting deadlines in advance for the completion of each phase.

| | | |
|---|---|---|
| 1. | Each $T_k$: | Select and reveal a set of parameter values S to be used in the election and interactively prove that S conforms to certain specifications. |
| 2. | Each $V_\ell$: | Select and reveal an unmarked ballot $B_\ell$ consisting of an encrypted yes vote and an encrypted no vote in random order and interactively prove that $B_\ell$ is of this form. |
| 3. | Each $V_\ell$: | Select one vote as the actual vote on the ballot $B_\ell$. |
| 4. | Each $T_k$: | Release $\tau_k$ — a composition of the values received by $T_k$ and interactively proof that $\tau_k$ is this composition. |

The tally of an election is $\Gamma(\tau_1, \ldots, \tau_K)$ where $\Gamma$ is a pre-determined function.

Figure 5.1: The election paradigm.

The four phases of an election correspond in a fairly natural way to aspects of actual elections. The first phase is an *announcement* phase in which the election is announced, perhaps dates are selected, and parameters and rules which will guide the election are set. The second phase consists of voter *registration* in which eligible voters identify themselves and register for the election. The third phase is the actual *voting* phase in which voters select their votes. The fourth phase is the *tally* phase in which the election tally is compiled and revealed. Notice that the four phases do not have to take place in rapid succession. In particular, registration may be done well before the actual voting, and the

decision of how (or whether) to cast a vote need not be made until the voting phase.

The basic election schema of section 3.4 is a special case of this paradigm. In the basic election schema, there is only one teller (called the government). In this case, the tally function $\Gamma$ can be simply the identity function. Such a government is, of course, able to determine how every voter votes in an election. It is, however, still useful to see how a government can convince voters of the correctness of a tally without releasing the actual votes.

In the more general "teller schema" to be presented next, each teller functions essentially as an autonomous "sub-government". However, the "tally-shares" released by the tellers are not meaningful until they are combined by $\Gamma$.

## 5.5 Votes and Ballots

Section 2.9 (Elementary Elections) presented the notion of using a random member of residue class 0 to denote a no vote and a random member of residue class 1 to denote a yes vote. This notion was maintained in section 3.4 (Basic Elections) where elections still required only a single government entity. Generalizing to the case of multiple tellers, however, requires a corresponding change to the notion of a vote.

Suppose we are to hold an election with $K$ tellers such that the election can withstand up to $F$ teller faults. We embed the basic election schema of section 3.4 within a $(J, K)$ Shamir secret sharing scheme where $J = K - F$. Let $r$ be a fixed prime greater than the number of eligible voters and suppose each teller $T_k$ has selected election parameters $(n_k, y_k)$ such that $(r, n_k, y_k)$ is consonant.

**Definition** Given a prime $r$, we say that a vector

$$P = \langle (n_1, y_1), (n_2, y_2), \ldots, (n_\kappa, y_\kappa) \rangle$$

of parameters is *r-proper* if for every $k$, $(r, n_k, y_k)$ is a prime consonant triple.

This condition is satisfied whenever $y_k \in \overline{\mathbf{Z}^{\mathbf{r}}_{\mathbf{n_k}}}$ (see theorem 2.18).

**Definition** Given a prime $r$ and an $r$-proper parameter vector

$$P = \langle (n_1, y_1), (n_2, y_2), \ldots, (n_\kappa, y_\kappa) \rangle,$$

define

$$\mathbf{Z}^*_{\mathbf{P}} = \mathbf{Z}^*_{\mathbf{n_1}} \times \mathbf{Z}^*_{\mathbf{n_2}} \times \cdots \times \mathbf{Z}^*_{\mathbf{n_\kappa}}$$

to be the direct product of the groups $\mathbf{Z}^*_{\mathbf{n_k}}$. As the direct product of groups, $\mathbf{Z}^*_{\mathbf{P}}$ is itself a group.

**Definition** Given a prime $r$ and an $r$-proper parameter vector

$$P = \langle (n_1, y_1), (n_2, y_2), \ldots, (n_\kappa, y_\kappa) \rangle,$$

define

$$\mathbf{D}^{\mathbf{r}}_{\mathbf{P}} = \mathbf{D}^{\mathbf{r}}_{(\mathbf{n_1},\mathbf{y_1})} \times \mathbf{D}^{\mathbf{r}}_{(\mathbf{n_2},\mathbf{y_2})} \times \cdots \times \mathbf{D}^{\mathbf{r}}_{(\mathbf{n_\kappa},\mathbf{y_\kappa})}$$

to be the direct product of the decomposition sets $\mathbf{D}^{\mathbf{r}}_{(\mathbf{n_k},\mathbf{y_k})}$ defined in section 2.6. Of course, since (by lemma 2.25) each $\mathbf{D}^{\mathbf{r}}_{(\mathbf{n_k},\mathbf{y_k})}$ is a group, the direct product $\mathbf{D}^{\mathbf{r}}_{\mathbf{P}}$ is also a group with the straightforward generalization of $\oplus$ as its operation.

As in section 2.6, we are able to define an evaluation function

$$\Psi : \mathbf{D}^{\mathbf{r}}_{\mathbf{P}} \to \mathbf{Z}^*_{\mathbf{P}}.$$

**Definition** Let $P = \langle (n_1, y_1), (n_2, y_2), \ldots, (n_\kappa, y_\kappa) \rangle$ be an $r$-proper parameter vector. Define $\Psi : \mathbf{D}^{\mathbf{r}}_{\mathbf{P}} \to \mathbf{Z}^*_{\mathbf{P}}$ by

$$\begin{aligned}&\Psi(\langle [c_1, x_1], [c_2, x_2], \ldots, [c_\kappa, x_\kappa] \rangle) \\ &\quad = \langle y_1^{c_1} x_1^r \bmod n_1, y_2^{c_2} x_2^r \bmod n_2, \ldots, y_\kappa^{c_\kappa} x_\kappa^r \bmod n_\kappa \rangle.\end{aligned}$$

As a direct product of homomorphisms, $\Psi$ is a homomorphism.

**Definition** Given a $D \in \mathbf{D}^{\mathbf{r}}_{\mathbf{P}}$, $\Psi(D)$ is called the *evaluation* of $D$. Given a $W \in \mathbf{Z}^*_{\mathbf{P}}$, a $D \in \mathbf{D}^{\mathbf{r}}_{\mathbf{P}}$ such that $\Psi(D) = W$ is called a *decomposition* of $W$.

As in the scalar case, every $D \in \mathbf{D}^{\mathbf{r}}_{\mathbf{P}}$ has a unique evaluation, but a $W \in \mathbf{Z}^*_{\mathbf{P}}$ may have multiple decompositions.

**Definition** Given an $r$-proper parameter vector

$$P = \langle (n_1, y_1), (n_2, y_2), \ldots, (n_\kappa, y_\kappa) \rangle$$

and a vector $W = \langle w_1, w_2, \ldots, w_\kappa \rangle \in \mathbf{Z}^*_{\mathbf{P}}$, the *disclosure* of $W$ (with respect to $r$ and $P$) denoted by $[\![W]\!]_{(r,P)}$ (or simply $[\![W]\!]$ when $r$ and $P$ are understood) is the $\kappa$-component vector $C = \langle c_1, c_2, \ldots, c_\kappa \rangle$ where each $c_k = [\![w_k]\!]_{(r,n_k,y_k)}$. Note that since each $(r, n_k, y_k)$ is only required to be prime consonant and *not* required to be perfect consonant, it is possible that $[\![w_k]\!]_{(r,n_k,y_k)} = \infty$.

### 5.5.1 Votes

**Definition** Given a vector $C = \langle c_1, c_2, \ldots, c_\kappa \rangle$ and a threshold $J$, let $\|C\|_J$ (or simply $\|C\|$ when $J$ is understood) denote the value at 0 of the minimum degree polynomial passing through the points $(k, c_k)$ if this polynomial is of degree less than $J$. That is, $\|C\|$ is the secret defined by the shares $c_1, c_2, \ldots, c_\kappa$. If any of the $c_k = \infty$ or if the minimum degree polynomial passing through the points $(k, c_k)$ is of degree $J$ or greater, then $\|C\|_J = \infty$.

**Definition** Given $J$, $K$, with $0 < J \leq K$, a prime $r$, and a $K$-component, $r$-proper parameter vector $P$, we say the *type* of a $W \in \mathbf{Z}_\mathbf{P}^*$ is the secret defined by the disclosure of $W$, i.e. $\text{type}(W) = \|[\![W]\!]\|$. If any element of the disclosure of $W$ is undefined ($\infty$), or if the disclosure of $W$ does not define a unique secret, then $\text{type}(W) = \infty$.

**Definition** Given $J$, $K$, with $0 < J \leq K$, a prime $r$, and a $K$-component, $r$-proper parameter vector $P$, a *vote* is defined to be a $K$-component vector of integers $W = \langle w_1, w_2, \ldots, w_K \rangle$ such that $\|[\![W]\!]\|$ is finite. A vote of type 0 is called a *no vote*, and a vote of type 1 is called a *yes vote*. A vote which is either a no vote or a yes vote is said to be *valid*. Let $\Omega(J, K, r, P)$ (or simply $\Omega$ when $J$, $K$, $r$, and $P$ are understood) denote the set of votes with respect to $J$, $K$, $r$, and $P$.

**Definition** Given two votes $U = \langle u_1, u_2, \ldots, u_K \rangle$ and $V = \langle v_1, v_2, \ldots, v_K \rangle$ in $\Omega(J, K, r, \langle (n_1, y_1), (n_2, y_2), \ldots, (n_K, y_K) \rangle)$, we define their product $U \circ V$ to be the vector $U \circ V = \langle u_1 v_1, u_2 v_2, \ldots, u_K v_K \rangle$ where each product $u_k v_k$ is taken modulo $n_k$.

We begin by giving generalizations of lemmas 2.8 and 2.9 of section 2.3.

**Lemma 5.1** *If $U = \langle u_1, u_2, \ldots, u_K \rangle$ and $V = \langle v_1, v_2, \ldots, v_K \rangle$ are votes of type $u$ and $v$, respectively, then the product $U \circ V$ is a vote of type $u + v \bmod r$.*

*Proof:*

By definition, $U$ is of type $u$ if and only if the secret defined by the disclosure of $U$ is $u$. Similary, $V$ is of type $v$ if and only if $\|[\![V]\!]\| = v$. Now

$$[\![U \circ V]\!] = \langle [\![u_1 v_1]\!]_{(r,n_1,y_1)}, [\![u_2 v_2]\!]_{(r,n_2,y_2)}, \ldots, [\![u_K v_K]\!]_{(r,n_K,y_K)} \rangle.$$

But, by lemma 2.8, this is equal to

$$\langle \quad [\![u_1]\!]_{(r,n_1,y_1)} + [\![v_1]\!]_{(r,n_1,y_1)} \bmod r,$$

$$[\![u_2]\!]_{(r,n_2,y_2)} + [\![v_2]\!]_{(r,n_2,y_2)} \bmod r,$$
$$\ldots,$$
$$[\![u_K]\!]_{(r,n_K,y_K)} + [\![v_K]\!]_{(r,n_K,y_K)} \bmod r\rangle.$$

That is to say, the disclosure of $U \circ V$ is the componentwise sum of the disclosures of $U$ and $V$. By lemma 4.1, Shamir's secret sharing scheme is $(+,+)$-homomorphic. Thus, the sum of the shares

$$\begin{aligned}\langle \quad & [\![u_1]\!]_{(r,n_1,y_1)} + [\![v_1]\!]_{(r,n_1,y_1)} \bmod r,\\ & [\![u_2]\!]_{(r,n_2,y_2)} + [\![v_2]\!]_{(r,n_2,y_2)} \bmod r,\\ & \ldots,\\ & [\![u_K]\!]_{(r,n_K,y_K)} + [\![v_K]\!]_{(r,n_K,y_K)} \bmod r\rangle\end{aligned}$$

constitute shares of the sum $u + v$ of the original secrets. The secret domain, however, is limited to $\mathbf{Z_r}$. Hence, $U \circ V$ is a vote of type $u + v \bmod r$. ∎

**Definition** Given a vote $W = \langle w_1, w_2, \ldots, w_K\rangle$ in

$$\Omega(J, K, r, \langle (n_1, y_1), (n_2, y_2), \ldots, (n_K, y_K)\rangle),$$

we define their product $W^{-1}$ to be the vector $W^{-1} = \langle w_1^{-1}, w_2^{-1}, \ldots, w_K^{-1}\rangle$ where each inverse $w_k^{-1}$ is taken modulo $n_k$.

**Lemma 5.2** *If $W = \langle w_1, w_2, \ldots, w_K\rangle$ is a vote of type $t$, then $W^{-1}$ is a vote of type $-t \bmod r$.*

*Proof:*

Since $W$ is a vote, each $w_k$ is relatively prime to its corresponding $n_k$. Thus, $W^{-1}$ is well-defined. Let $C = \langle c_1, c_2, \ldots, c_K\rangle = [\![W]\!]$ be the disclosure of $W$.

$$[\![W^{-1}]\!] = -C \bmod r = \langle -c_1 \bmod r, -c_2 \bmod r, \ldots, -c_K \bmod r\rangle$$

since, by lemma 2.9, if $w_k$ is of class $c_k$ then $w_k^{-1}$ is of class $-c_k$. Also, the properties of polynomials readily imply that $\|-C\| = -\|C\| \bmod r$. Thus, $W^{-1}$ is a vote of type $-t \bmod r$. ∎

It is now easy to show that the set $\Omega$ is a group.

**Lemma 5.3** *Given $J$, $K$, with $0 < J \leq K$, a prime $r$, and a $K$-component, $r$-proper parameter vector $P$, the set $\Omega(J, K, r, P)$ is a commutative group under the $\circ$ operation.*

*Proof:*

The element $\overline{1} = \langle 1, 1, \ldots, 1 \rangle$ is clearly an identity for $\Omega$.

Lemma 5.1 shows that $\Omega$ is closed under multiplication.

By lemma 5.2, $W^{-1}$ is a well-defined vote, and clearly $W \circ W^{-1} \equiv W^{-1} \circ W \equiv \overline{1}$, so $W^{-1}$ is an inverse of $W$. Finally, associativity and commutativity are inherited from integer multiplication. ∎

We are now able to give a generalization of lemma 2.10 of section 2.3.

**Lemma 5.4** *Two votes $U = \langle u_1, u_2, \ldots, u_K \rangle$ and $V = \langle v_1, v_2, \ldots, v_K \rangle$ are of the same type (with respect to a given parameter vector) if and only if there exists a vote $W = \langle w_1, w_2, \ldots, w_K \rangle$ of type 0 such that $U \equiv V \circ W$.*

*Proof:*

Assume there exists a vote $W$ of type 0 such that $U = V \circ W$. Then, by lemma 5.1, the type of $U$ is equal to the sum of the types of $V$ and $W$. But the $W$ is of type 0. Thus, the type of $U$ must equal the type of $V$.

Conversely, let $W = U \circ V^{-1}$. Since, $\Omega$ is a commutative group, $U = V \circ W$. If $U$ and $V$ are both of type $t$, then, by lemma 5.2, $V^{-1}$ is a vote of type $-t \bmod r$. Thus, by lemma 5.1, $W = U \circ V^{-1}$ is a vote of type 0. ∎

## 5.5.2 Decompositions of Votes

**Definition** Let $\Delta \subseteq \mathbf{D}_{\mathbf{P}}^{\mathbf{r}}$ be defined by $\Delta = \{D \in \mathbf{D}_{\mathbf{P}}^{\mathbf{r}} : \Psi(D) \in \Omega\}$. In other words, $\Delta$ is the set of $D = \langle [c_1, x_1], [c_2, x_2], \ldots, [c_K, x_K] \rangle \in \mathbf{D}_{\mathbf{P}}^{\mathbf{r}}$ such that $\| \langle c_1, c_2, \ldots, c_K \rangle \|$ is finite.

**Lemma 5.5** *$\Delta$ is a subgroup of $\mathbf{D}_{\mathbf{P}}^{\mathbf{r}}$.*

*Proof:*

The identity of $\mathbf{D}_{\mathbf{P}}^{\mathbf{r}}$ is the element $\langle [0,1], [0,1], \ldots, [0,1] \rangle$, which describes the secret value 0. Thus, the identity of $\mathbf{D}_{\mathbf{P}}^{\mathbf{r}}$ is in $\Delta$. If $D_1, D_2 \in \Delta$, then $\Psi(D_1 \oplus D_2) = \Psi(D_1) \circ \Psi(D_2)$, and by definition, each $\Psi(D_i)$ is in $\Omega$. Hence, $\Psi(D_1 \oplus D_2) \in \Omega$, so $D_1 \oplus D_2 \in \Delta$. Finally, if $D \in \Delta$, then $\Psi(\ominus D) = \Psi(D)^{-1} \in \Omega$, so $(\ominus D) \in \Delta$. ∎

Thus, in particular, the binary operation $\ominus$ defined by

$$D_1 \ominus D_2 = D_1 \oplus (\ominus D_2)$$

is well-defined on $\Delta$; and given $D_1$ and $D_2$ such that $\Psi(D_1) = W_1$ and $\Psi(D_2) = W_2$,

$$\begin{aligned} \text{type}(\Psi(D_1 \ominus D_2)) &= \text{type}(\Psi(D_1) \circ \Psi(D_2)^{-1}) \\ &= \text{type}(W_1 \circ W_2^{-1}) \\ &= \text{type}(W_1) - \text{type}(W_2). \end{aligned}$$

Therefore, the difference between the types of $W_1$ and $W_2$ can be shown by revealing $D_1 \ominus D_2$ which is a decomposition of $W_1 \circ W_2^{-1}$.

This property, which is a consequence of lemma 5.4, can, in analogy with lemma 2.10, be used as the basis for an interactive proof almost identical to that of section 3.1.3 to show that two votes are of the same type.

**Remark** It is possible to define votes as a single integer rather than a vector. The Chinese Remainder Theorem defines an isomorphism between the group $\mathbf{Z}_P^*$ where $P = \langle (n_1, y_1), (n_2, y_2), \ldots, (n_K, y_K) \rangle$ and the group $\mathbf{Z}_N^*$ where $N = \prod_k n_k$. All of the operations on votes can be redefined to conform to this viewpoint without any loss of functionality. Although there are some advantages to defining a vote to be a single integer, the size of the integer is, of course, comparable to the size of the vector of integers used. For purposes of clarity, the components have been kept separate here.

### 5.5.3 Ballots

**Definition** A capsule consisting of a pair of votes is called a *ballot.*

**Definition** A ballot consisting of a no vote and a yes vote (in either order) is called a *valid ballot.*

**Definition** A ballot consisting of two no votes (i.e two votes of type 0) is called a *zero ballot.*

**Definition** The *type* of a ballot $B = (W_1, W_2)$ is the ordered pair

$$(\text{type}(W_1), \text{type}(W_2)).$$

Two ballots are said to be of *equivalent type* if their types are the same or the reverse of each other.

We also define two quotient operators on ballots.

**Definition** If $B = (W_1, W_2)$ and $B' = (W_1', W_2')$ are ballots, then the *forward quotient* $B'/B$ is defined to be the ballot $(W_1' \circ W_1^{-1}, W_2' \circ W_2^{-1})$, and the *reverse quotient* $B' \backslash B$ is defined to be the ballot $(W_1' \circ W_2^{-1}, W_2' \circ W_1^{-1})$.

Note that two ballots $B$ and $B'$ are of equivalent type if and only if at least one of $B/B'$ and $B\backslash B'$ is a zero ballot. In particular, if $B$ and $B'$ are both valid ballots, then exactly one of $B/B'$ and $B\backslash B'$ is a zero ballot.

## 5.6 A Verifiable Secret-Ballot Election Schema

Here, at last, are the protocols for the full fault-tolerant verifiable secret-ballot election schema. With the machinery that has been constructed, the schema can be described fairly simply.

Besides the sets $\mathcal{T}$ of tellers and $\mathcal{V}$ of voters, we introduce two additional sets of participants. These additional sets may be drawn from $\mathcal{T} \cup \mathcal{V}$ or they may consist of independent participants.

The first of these additional sets is the set of *bit generators* denoted by $\mathcal{G}$. Let $M = |\mathcal{G}|$. The sole responsibility of the bit generators is to generate random bits at various times to facilitate interactive proofs. If a single trusted random source (known as a *beacon* — [Rabi83a]) is available, then $\mathcal{G}$ can be the singleton set containing just this beacon. In general, $\mathcal{G}$ should be chosen to be a set of random sources such that each participant is confident in the integrity and the unpredictability of *at least one* member of $\mathcal{G}$.

The second additional set is the set of *inspectors* denoted by $\mathcal{I}$. Let $I = |\mathcal{I}|$. The mission of the inspectors will be to engage the tellers in a set of interactive proofs to ensure the validity of the election parameters. As with the bit generators, the inspector set $\mathcal{I}$ should be chosen such that each participant is confident in the integrity of *at least one* member of $\mathcal{I}$. The inspector set $\mathcal{I}$ may consist of the bit generators $\mathcal{G}$, the tellers $\mathcal{T}$ themselves (challenging each other), or the entire set of voters $\mathcal{V}$. The reasons for selecting one of these sets over another are dictated by the assumed capabilities of the agents and by efficiency considerations. This is discussed further in section 5.11. It should be emphasized that the inspectors are *not* able to "oversee" an entire election. They (even as a set) are not able to read voters' private votes nor obtain any special information about the election parameters from the tellers. Their role

is quite limited and their introduction is merely a notational convenience which adds flexibility to the election schema.

We assume that a prime $r$ which is larger than the number of eligible voters has been fixed in advance and that a security parameter $N$ has also been selected.

The election schema consists of four basic phases according to the paradigm described in section 5.4. Within each of these phases there may be several sub-phases.

In the announcement phase, each teller $T_k \in \mathcal{T}$ randomly selects a pair $(n_k, y_k)$ such that the triple $(r, n_k, y_k)$ is prime consonant. Each teller uses the interactive proof technique of section 3.3.2 to convince each member of the inspector set $\mathcal{I}$ that $(r, n_k, y_k)$ is, in fact, prime consonant (see figure 5.2).

In the registration phase, each voter $V_\ell \in \mathcal{V}$ that wishes to vote prepares a valid ballot as described in section 5.5.3. A slightly generalized version of the interactive proof of section 3.1.3 is then used to demonstrate the validity of the ballot (see figure 5.3).

In the voting phase, each voter $V_\ell \in \mathcal{V}$ that has satisfactorily completed its interactive proof may then cast a vote by designating one of the two votes on its ballot as the vote to be cast (see figure 5.4).

In the tally phase, the $(\circ)$ product of the votes cast is computed. (These votes are completely public — although the secrets they represent are not — and the $(\circ)$ product can be computed by anyone and everyone.) The type of this product is the tally of the election. Each teller $T_k \in \mathcal{T}$ releases the type of its component of this vote product. An interactive proof using the technique of section 3.1.2 is used by each teller to demonstrate the class of its component (see figure 5.5).

Once a sufficient number of tellers have produced these "tally-shares", the overall tally can be computed as the secret defined by the set of tally-shares.

The tally function $\Gamma$ is defined to be the secret determined by any $J$ or more values $\tau_k$ for which $check_T(k) = \text{good}$.

(A) Each teller $T_k \in \mathcal{T}$ uniformly selects an $n_k$ with $|n_k| = N$ such that the pair $(r, n_k)$ is exact consonant (with error probability at most $1/(K2^N)$) as in lemma 2.27. Each teller then uniformly selects a $y \in \mathbf{Z}^*_{\mathbf{n_k}}$ such that the triple $(r, n_k, y_k)$ is exact consonant as in lemma 2.28. Teller $T_k$ posts the pair $(n_k, y_k)$ as its election parameters.

(B) Let $\eta_1 = \lceil (\log_2 |\mathcal{I}|)(\log_2 \log_2 |\mathcal{T}|)N \rceil$ and let $\eta_2 = \lceil (\log_2 |\mathcal{T}|)N \rceil$.
Each inspector $I_s$ selects, for each teller $T_k$, $\eta_2$ values $c_i \in \{0, 1\}$ and $x_i$ uniformly from $\mathbf{Z}^*_{\mathbf{n}}$ and posts the $\eta_2$ values $w_i = \psi([c_i, x_i]) = y_k^{c_i} x_i^r \bmod n_k$.
With each $w_i$, inspector $I_s$ randomly selects $\eta_1$ additional values $\alpha_{i,j}$ and $\beta_{i,j}$, all relatively prime to $n_k$ and posts $\eta_1$ capsules each consisting of (in random order) the pair $u_{i,j} = \psi([0, \alpha_{i,j}]) = \alpha_{i,j}^r \bmod n_k$ and $v_{i,j} = \psi([1, \beta_{i,j}]) = y_k \beta_{i,j}^r \bmod n_k$.

(C) Each teller $T_k$, for each inspector $I_s$, selects $\eta_1 \eta_2$ random bits $b_{i,j}$, $1 \le i \le \eta_2$ and $1 \le j \le \eta_1$.

(D) For each pair $(i, j)$ such that $b_{i,j} = 0$, $I_s$ releases the values $[0, \alpha_{i,j}]$ and $[1, \beta_{i,j}]$.
For each pair $(i, j)$ such that $b_{i,j} = 1$, $I_s$ selects and posts

$$w_{i,j} = \begin{cases} u_{i,j} & \text{if } c_i = 0; \\ v_{i,j} & \text{if } c_i = 1. \end{cases}$$

Let $[c_{i,j}, x_{i,j}]$ be a decomposition of $w_{i,j}$. That is, $[c_{i,j}, x_{i,j}]$ is either $[0, \alpha_{i,j}]$ or $[1, \beta_{i,j}]$. Inspector $I_s$ then releases the decomposition of $w_{i,j} w_i^{-1}$ given by

$$[c_{i,j}, x_{i,j}] \ominus [c_i, x_i] = \begin{cases} [c_{i,j} - c_i, x_{i,j} x_i^{-1} \bmod n_k, & \text{if } c_{i,j} - c_i \ge 0; \\ [c_{i,j} - c_i + r, x_{i,j} x_i^{-1} y_k^{-1} \bmod n_k, & \text{if } c_{i,j} - c_i < 0. \end{cases}$$

(E) Each teller $T_k$ then releases $[\![w_i]\!]_{(r, n_k, y_k)}$ for all $i$.

(F) Each challenger $I_s$ releases $[c_i, x_i]$ for all $i$.

Figure 5.2: Phase 1: The announcement phase.

(A) Let $\eta_3 = \lceil (\log_2 |\mathcal{V}|)N \rceil$.
Each voter $V_\ell$ prepares as (unmarked) ballots, $\eta_3 M + 1$ capsules $B_i$ each consisting of a no vote and a yes vote as described in section 5.5. One of these capsules is designated as the *master* ballot $B_0$, and $\eta_3$ of these capsules are associated with each of the $M$ bit generators.

(B) Each bit generator $G_m$ produces, for each voter $V_\ell$, bits $b_i$, $1 \leq i \leq \eta_3$.

(C) For all $i$ such that $b_i = 0$, voter $V_\ell$ decomposes its associated $B_i$. For all $i$ such that $b_i = 1$, voter $V_\ell$ decomposes either $B_i/B_0$ or $B_i \backslash B_0$ — whichever of the two is a zero ballot.

Figure 5.3: Phase 2: The registration phase.

Each voter $V_\ell$ designates one of the two votes of its master ballot $B_0$ as the actual vote to be cast in the election.

Figure 5.4: Phase 3: The voting phase.

(A) Let $\eta_4 = \lceil (\log_2 |\mathcal{T}|)N \rceil$.
Each teller $T_k$ computes the set of "correct" voters $V_\ell$ for which $check_V(\ell) = \text{good}$ and then computes $W_k$ which is the product modulo $n_k$ of the $k^{\text{th}}$ component of all votes cast by the correct voters.
Each teller $T_k$ reveals $\tau_k = [\![W_k]\!]_{(r,n_k,y_k)}$.
Each teller $T_k$ selects $\eta_4 M$ random values $s_i \in \mathbf{Z}^*_{\mathbf{n_k}}$ and reveals $S_i \equiv_{n_k} s_i^r$. From these $\eta_4 M$ values of $S_i$, $\eta_4$ of the $S_i$ are associated with each of the $M$ bit generators.

(B) Each bit generator $G_m$ produces, for each teller $T_k$, bits $b_i$ associated with the $S_i$.

(C) For all $i$ such that $b_i = 0$, teller $T_k$ reveals its associated $s_i$. For all $i$ such that $b_i = 1$, teller $T_k$ reveals $s_i x_k$, where $x_k$ is an $r^{\text{th}}$ root of $W_k y_k^{-\tau_k}$.

Figure 5.5: Phase 4: The tally phase.

## 5.7 The function *check*

In order to gain confidence that the tally is correct, it is necessary to evaluate the *check* function. The check function does not require computation of residue classes or examination of challenges, but it does require that each vote cast be checked for consistency. The function *check* is defined in terms of a function $check_T$. The function $check_T$ is in turn defined in terms of $check_I$ and $check_V$.

### 5.7.1 $check_I$

The value of the function $check_I(s,k)$ is defined to be good if and only if

1. In phase 1(B), inspector $I_s$ has posted up to $\eta_2$ complete sets each containing an element $w_i$ and $\eta_1$ capsules. Each $w_i$ and the elements of each capsule must all be relatively prime to $n_k$.

2. In phase 1(D), for each capsule with which teller $T_k$ associates a 0, inspector $I_s$ posts a decomposition showing one component to be of $\mathcal{RC}[0]$ and the other to be of $\mathcal{RC}[1]$.

3. In phase 1(D), for each capsule with which teller $T_k$ associated a 1, inspector $I_s$ chooses a component $w_{i,j}$ and gives a decomposition of $w_{i,j}w_i^{-1}$. Each such decomposition must show $w_{i,j}w_i^{-1}$ to be in $\mathcal{RC}[0]$.

4. In phase 1(F), inspector $I_s$ posts a decomposition of $w_i$.

### 5.7.2 $check_V$

The value of the function $check_V(\ell)$ is defined to be good if and only if

1. In phase 2(A), voter $V_\ell$ posts a master ballot and $M$ sets consisting of $\eta_3$ ballots each. Every element of every component of every ballot posted must be relatively prime to its associated modulus $n_i$. Let $B$ denote this master ballot.

2. In phase 2(C), for each ballot with which the corresponding bit generator has associated a 0, voter $V_\ell$ posts a complete decomposition of the ballot. These decompositions must show these ballots to be valid ballots.

3. In phase 2(C), for each ballot $B'$ with which the corresponding bit generator has associated a 1, voter $V_\ell$ posts a complete decomposition of either $B'/B$ or $B'\backslash B$. These decompositions must show these quotients to be zero ballots.

4. In phase 3, voter $V_\ell$ designates one of the two votes of its master ballot as the vote to be cast.

### 5.7.3 $check_T$

The value of the function $check_T(k)$ is defined to be good if and only if

1. In phase 1(A), teller $T_k$ posts a pair $(n_k, y_k)$ such that $y_k$ is relatively prime to $n_k$.

2. In phase 1(C), teller $T_k$ associates a bit with each capsule presented to it by an inspector.

3. In phase 1(E), for each $w_i$ presented by an inspector $I_s$ for which the value of $check_I(s,k)$ is good, teller $T_k$ associates a bit designating to which of $\mathcal{RC}[0]$ and $\mathcal{RC}[1]$ $w_i$ belongs.

4. No inspector for which $check_I(s,k) =$ good posts a decomposition in phase 1(F) which contradicts a residue class claimed by teller $T_k$.

5. In phase 4(A), teller $T_k$ posts a "tally-share" $\tau_k$.

6. In phase 4(A), teller $T_k$ posts $\eta_4 M$ values $w$, each relatively prime to $n_k$.

7. In phase 4(C), for each value $w$ with which the corresponding bit generator has associated a 0, teller $T_k$ posts an $r^{\text{th}}$ root of $w$.

8. In phase 4(C), for each value $w$ with which the corresponding bit generator has associated a 1, teller $T_k$ posts an $r^{\text{th}}$ root of $W_k y^{-\tau_k} w$, where $W_k$ is the product modulo $k$ of the $k^{\text{th}}$ component of all votes cast by voters $V_\ell$ for which $check_V(\ell) =$ good.

### 5.7.4 *check*

The value of *check* is defined to be good if and only if $check_T(k) =$ good for at least $J$ tellers $T_k$.

We shall show in section 5.9 that the probability that *check* = good while the tally is incorrect decreases exponentially as the security parameter $N$ increases.

## 5.8 Time and Space Requirements

It is important to analyze the resources used in the election schema of section 5.6. Recall that $K = |\mathcal{T}|$ is the number of tellers, $L = |\mathcal{V}|$ is the number of voters, $M = |\mathcal{G}|$ is the number of bit generators, $I = |\mathcal{I}|$ is the number of inspectors, and $N$ is the security parameter.

### 5.8.1 Time Requirements

Although the election schema has many complicated aspects, almost all of the computation times are bounded by the cost of performing gcd operations. When selecting a random $x \in \mathbf{Z}_n^*$, $\gcd(x, n)$ is computed to verify that $x$ is, in fact, relatively prime to $n$. When $N = |n|$, such a gcd requires $\mathcal{O}(N)$ $N$-bit multiplication/division operations. Once such an $x$ has been chosen, computing $y^c x^r \bmod n$ (for $0 \leq c < r$) requires only $\mathcal{O}(\log r)$ $N$-bit multiplication/division operations. Thus (since $r < n$), the cost of selecting a random member of residue class $c$ is dominated by the cost of the gcd and is $\mathcal{O}(N)$ times the cost of performing an $N$-bit multiplication or division.

#### Voter Protocols

In the registration phase of the election schema, each voter prepares $\mathcal{O}(\eta_3 M) = \mathcal{O}((\log L)MN)$ ballots each consisting of $K$ pairs of integers selected from specific classes. Thus, each voter requires $\mathcal{O}((\log L)KMN^2)$ $N$-bit multiplications and divisions to complete phase 2.

Phase 3 only requires each voter to designate one bit of information, and phase 4 requires no voter participation. Thus, the total time required for each voter protocol is the time needed to perform

$$\mathcal{O}((\log L)KMN^2)$$

$N$-bit multiplications and/or divisions. That is, the number of $N$-bit multiplications and divisions required of a voter is proportional to the square of the security parameter ($N$), the number of tellers ($K$), the number of bit generators ($M$), and the logarithm of the number of voters ($L$).

### Teller Protocols

The first step required of each teller is the selection of a pair $(n, y)$ such that $(r, n, y)$ is prime consonant. The method of lemmas 2.27 and 2.28 require the selection of primes $p$ and $q$ such that $r \mid (p-1)$, $r^2 \nmid (p-1)$, and $r \nmid (q-1)$. Such a $p$ can be randomly chosen by selecting an $A$ of size roughly $\mathcal{O}(n)$ and a $B$ uniformly selected among integers from 1 to $r-1$, inclusive, and computing $p = Ar^2+Br+1$. The general form of the prime number theorem used in lemma 2.27 (see [Kran86]) ensures that such integers have roughly the same density of primes as integers in general. Thus, approximately 1 out of $N$ of such integers $p$ are prime. Probabilistic primality tests can be used to test such $p$. These tests can verify primality with confidence $1 - 2^{\eta}$ using $\mathcal{O}(\eta N)$ $N$-bit multiplications and divisions. Composite $N$ can be discovered with $\mathcal{O}(N)$ expected $N$-bit multiplications and divisions. Thus, the total expected time to find such a $p$ is $\mathcal{O}(\eta N + N^2))$ times the time for an $N$-bit multiplication/division, and the probability of producing a "false" prime is bounded by $T/2^{\eta}$, where $T$ is the number of trials and is $\mathcal{O}(N)$. Similarly, a prime $q$ of the required form can be generated by selecting an $A$ of roughly the size of $n$ and a $B$ uniformly selected among integers from 2 to $r-1$, inclusive, and computing $q = Ar + B$. Once again, integers of this form have roughly the same density of primes as integers in general. Thus, such a $q$ can be found in expected time $\mathcal{O}(\eta N+N^2))$ times the time for an $N$-bit multiplication/division with a comparable error bound. Thus, selecting an exact consonant pair with error probability less than $1/(K2^N)$ is accomplished by setting $\eta = \lceil (\log_2 K)N(\log_2 N) \rceil$ and requires the same time as is required to perform $\mathcal{O}((\log K)N^2(\log N)))$ $N$-bit multiplication/division operations.

Next, given $n$ such that $(r, n)$ is exact consonant, lemma 2.28 asserts that a $(r-1)/r$ fraction of the $y \in \mathbf{Z}_n^*$ are not in $\mathbf{Z}_n^r$. Thus, the expected number of such $y$ which must be tested is constant. Such a test requires a gcd and (by Lemma 2.23) a modular exponentiation. Thus, an expected $\mathcal{O}(N)$ $N$-bit

multiplications/divisions are used to generate the required $y$. Hence, selection of a suitable $(n, y)$ pair requires an expected $O((\log K)N^2(\log N))$ $N$-bit multiplications and divisions.

The remaining obligations of the tellers consist almost entirely of determining the residue classes of various integers. Each teller must do this once for every challenge it receives (up to $\eta_2 = \lceil(\log_2 K)\rceil N$ challenges per inspector) and once more to tally its vote-shares by computing the residue class of their product. Thus, up to $O(I(\log K)N)$ decodings (class determinations) are required of each teller.

The teller challenges are constrained to be elements of either $\mathcal{RC}[0]$ or $\mathcal{RC}[1]$. By lemma 2.23, these two cases can be distinguished with a single modular exponentiation (requiring at most $O(N)$ $N$-bit multiplications and divisions). Therefore, the total number of $N$-bit multiplications and divisions necessary to answer the challenges is $O(I(\log K)N^2)$

Each teller must also compute the product of its component of the up to $L$ votes cast. This requires $O(L)$ $N$-bit multiplications and divisions.

The "big step – little step" method of section 2.5 can be used to compute the residue class of the single value which gives the teller's share of the tally. This requires $O(\sqrt{r}\log r)$ $N$-bit multiplications and divisions.

Finally, the interactive proof that the teller's share of the tally is correct requires $O(\eta_4 M) = O(\log K)MN)$ $N$-bit multiplications, divisions, and $r^{\text{th}}$ power computations. Thus, the total number of $N$-bit multiplications and divisions required is $O((\log K)MN(\log r))$.

Hence, the total time needed by each teller protocol is the time required to perform

$$O((\log K)N^2(\log N) + I(\log K)N^2 + L + \sqrt{r}\log r + (\log K)MN(\log r))$$

$N$ bit multiplication/division operations.

### Inspector Protocols

In the announcement phase of the schema, each inspector may challenge each teller. A challenge requires the preparation of $O(\eta_1) = O((\log I)(\log\log K)N)$ random members of known residue classes, and each inspector may challenge each of the $K$ tellers up to $\eta_2 = \lceil(\log_2 K)N\rceil$ times. Thus, each inspector

may prepare up to $O((\log I)K(\log K)(\log\log K)N^2)$ random members of known residue classes. Since each random selection requires $O(N)$ $N$-bit multiplications and divisions, each inspector requires

$$O((\log I)K(\log K)(\log\log K)N^3)$$

$N$-bit multiplication/divisions.

### Bit Generator Protocols

The bit generator protocols consist entirely of providing random bits when required to do so. Each bit generator must provide $\eta_3 = \lceil(\log_2 L)N\rceil$ bits to each of the $L$ voters to form the interactive proofs of vote validity. Also, each bit generator must provide $\eta_4 = \lceil(\log_2 K)N\rceil$ bits to each of the $K$ tellers to form the interactive proofs of the accuracy of the sub-tallies. Thus, each bit generator must produce a total of

$$(\lceil\log_2 L\rceil L + \lceil\log_2 K\rceil K)N$$

bits.

### Computing *check*

Almost all of the work required to compute *check* consists of computing greatest common divisors, inverses, and modular exponentiations. Each of these operations requires at most $O(N)$ $N$-bit multiplications and divisions. For each of the $I$ inspectors and each teller, $check_I(s,k)$ requires

$$O(\eta_1\eta_2) = O((\log I)(\log K)(\log\log K)N^2)$$

gcd, inverse, and modular exponentiation computations. For each of the up to $L$ voters, $check_V$ requires $O(\eta_3 KM) = O(K(\log L)MN)$ gcd, inverse, and modular exponentiation computations. For each of the $K$ tellers, $check_T$ requires $O(\eta_4 M+L)O((\log K)MN+L)$ gcd and modular exponentiation computations. Thus, computing check requires

$$O((I(\log I)K(\log K)(\log\log K)N^3 + KL(\log L)MN^2 + K(\log K)MN^2)$$

$N$-bit multiplication and division operations.

In particular, note that computing *check* requires time proportional to $L \log L$ rather than just $\log L$. This is explained by the fact that in order to verify the results of an election, one must "look over" the votes cast by *all* voters to ensure their validity.

### 5.8.2 Space Requirements

What is meant by space requirements here is the amount of permanent space which is required to hold the messages which must be posted. The amount of internal space required to perform the secret computations is quite small.

Each voter must post $\eta_3 M + 1$ ballots each consisting of two vectors which contain $K$ $N$-bit integers. Each voter must then post $\eta_3 M$ decompositions each consisting of two vectors which contain $K$ pairs consisting of an $N$-bit integer and a $(\log r)$-bit integer. Finally, each voter posts one additional bit — thereby indicating its vote. Thus, each of the $L$ voters may post up to $2KN(\eta_3+1)+2KM\eta_3(N+\log r)+1$ bits of information on its message board.

Each teller must post an $n$ and $y$ of $N$ bits each and for each inspector up to $\eta_1\eta_2$ bits and up to $\eta_2$ bits to answer each of the up to $I$ challenges. Later, each teller must post a $(\log r)$-bit tally-share and $2\eta_4 M$ $N$-bit integers to prove the validity of its tally-share. Thus, each of the $K$ tellers may post up to $2N + (\eta_1 + 1)\eta_2 + \log r + 2M\eta_4$ bits of information on its message board.

Each inspector may post up to $\eta_2$ $N$-bit challenge messages for each of the $K$ tellers. To interactively prove the validity of each of these challenges, each inspector posts up to and $\eta_1$ capsules each containing two $N$-bit messages. For each challenge, an inspector must then post decompositions of up to $2\eta_1$ values each consisting of a $(\log r)$-bit message and an $N$-bit message. Each inspector must then post a decomposition (again consisting of a $(\log r)$-bit message and an $N$-bit message) for each challenge. Thus, each of the $I$ inspectors may post up to $KN\eta_2 + 2KN\eta_1\eta_2 + 2K\eta_1\eta_2(N + \log r) + 2K\eta_2(N + \log r)$ bits of information on its message board.

The bit generators must each post a total of $\eta_3 L + \eta_4 K$ bits.

## 5.9 Correctness

In this section, we give the proof that the scheme $S$ presented in section 5.6 is a verifiable $(J, K, L)$ election scheme as defined in section 5.3.

The main tool in showing that the announced tally is correct is the use of interactive proofs. The basic idea of these interactive proofs is to force an agent who would produce fraudulent information to "outguess" the bit generators. As long as at least one of the generators and one of the inspectors is honest, this can be done successfully with only low probability.

The interactive proofs completed between the inspectors and the tellers give extremely high confidence that each $y$ is not an $r^{\text{th}}$ residue modulo $n$. By theorem 2.18, this implies that each $(r, n, y)$ triple is prime consonant and therefore that each $w \in \mathbf{Z}_n^*$ is expressible as $w \equiv_n y^c z$ with $z \in \mathbf{Z}_n^r$ for at most one integer $c$ such that $0 \leq c < r$.

The remaining interactive proofs give extremely high confidence that each vote cast is valid and that the tally-share released by each teller represents the actual residue class of the teller's component of the product of the votes cast.

Since, by lemma 5.1, the product of valid votes yields a vote whose type is the sum of the types of the votes, this type represents the number of yes votes cast in the election; and once at least $J$ tellers have released (and interactively proven) their tally-shares, this election tally can be computed. This is used to derive the following lemma.

**Lemma 5.6** *If at least one bit generator is honest and at least one inspector is honest, then with probability at least* $(1 - 1/(L2^N))$, $check_V(\ell) =$ good *if and only if voter* $V_\ell$ *votes properly.*

*Proof:*

If $V_\ell$ votes properly, then by construction $check_V(\ell) =$ good, and by the timing analysis of section 5.8.1, $V_\ell$ can complete its protocol in polynomially bounded time. If $V_\ell$ does not vote properly, then by definition $V_\ell$ posts some message which is inconsistent with its following either of its two prescribed protocols. Such a digression will be caught by the interactive proof of vote validity and therefore $check_V(\ell) =$ bad unless the interactive proof fails. But this occurs with probability at most $2^{-\eta_3} \leq 1/(L2^N)$. ∎

**Lemma 5.7** *If at least one bit generator is honest and at least one inspector is honest, then with probability at least* $(1 - 3/(K2^N))$, $check_T(k) =$ good *if and only if teller* $T_k$ *acts properly.*

*Proof:*

If $T_k$ acts properly, then by construction $check_T(k) =$ good unless teller $T_k$ selects a "false" prime as a factor of its first parameter $N_k$. By construction, this can only happen with probability less than $1/(K2^N)$. By the timing analysis of section 5.8.1, $T_k$ can complete its protocol in polynomially bounded time. Thus, the probability that teller $T_k$ acts properly and $check_T(k) =$ bad is at most $1/(K2^N)$.

If $T_k$ does not act properly, then the digression could occur in one of several places. $T_k$ could post a pair of election parameters $(n_k, y_k)$ such that $(r, n_k, y_k)$ is not consonant. $T_k$ could fail to answer one or more of the valid inspector challenges to its pair. $T_k$ could post an invalid tally-share. Finally, $T_k$ could fail to complete the interactive proof of the validity of its tally-share. If $T_k$ posts a pair which does not meet the consonance criterion, then it will not be able to correctly answer the challenges of the (at least one by assumption) honest inspector, and this will cause $check_T(k) =$ bad. If $(r, n_k, y_k)$ were not consonant, then $T_k$ could answer these challenges correctly with probability no greater than $2^{-\eta_2} \leq 1/(K2^N)$. If $T_k$ fails to answer a valid inspector challenge, then $check_T(k) =$ bad by construction. Similary, $check_T(k) =$ bad if $T_k$ fails to complete the interactive proof of the validity of the tally-share. If $T_k$ completes the interactive proof of the tally-share and $(r, n_k, y_k)$ is consonant, then the tally-share could only be invalid if its interactive proof failed which can happen with probability no more than $2^{-\eta_4} < 1/(K2^N)$. Thus, the probability that a teller $T_k$ acts improperly while $check_T(k)$ returns good is at most $2/(K2^N)$.

Therefore, with probability at least $1 - 3/(K2^N)$, $check_T(k) =$ good if and only if teller $T_k$ acts properly. ∎

By lemmas 5.6 and 5.7, if at least one bit generator is honest and at least one inspector is honest, then even if none of the tellers and none of the voters are proper, the joint probability is at most $4/2^N$ that *check* returns good and the election tally is not correct.

This is sufficient to give the following theorem.

**Theorem 5.8** *The election schema of section 5.6 is verifiable.*

*Proof:*

It can only be the case that *check* = good and the election tally is incorrect if at least one teller $T_k$ for which $check_T(k)$ = good releases an incorrect tally-share $\tau_k$ or if the vote of a voter which did not vote properly was counted towards the tally. By lemma 5.7 and since there are $K$ tellers, the probability of the former case is bounded by $3K/(K2^N) = 3/2^N$; and by lemma 5.6 and since there are at most $L$ voters, the probability of the latter case is bounded by $L/(L2^N) = 1/2^N$. Thus, the probability that the election tally is incorrect and $\doteq$good is bounded by $4/2^N = 1/2^{N-2}$, and therefore the election schema of section 5.6 is verifiable with confidence $1 - 1/2^{N-2}$.

Since the function $2^{N-2}$ grows faster than any polynomial function $P(N)$, the election schema of section 5.6 is verifiable. ∎

## 5.10 Security

It remains to be shown that compromising the votes of proper voters is hard. Unfortunately, significant lower bounds have been very scarce in theoretical computer science. The nature of public-key cryptography, in particular, makes it difficult to imagine a schema for which a super-polynomial lower bound on security would not immediately separate $\mathcal{P}$ and $\mathcal{NP}$. We must therefore be willing to settle for an equivalence result — a result that proves, by a rigorous reduction, that any breach of security would yield an efficient algorithm to solve a natural problem which is believed to be hard. Not surprisingly, the problem of compromising an election is closely related to the problem of deciding residuosity and the prime residuosity assumption.

It is easy to see that an oracle with the ability to decide residuosity can determine how individual voters voted in an election. It will, however, require far more effort to show that *any* oracle which, in possible collusion with voters, tellers, inspectors, and bit generators, can gain even a small advantage beyond chance at deciphering votes of honest voters, can itself be used to gain non-trivial information about deciding residues.

### 5.10.1 An Overview of the Reduction

The formal reduction is quite tedious; however, the basics are reasonably palatable. To begin with, we shall assume that we have a minimal set of honest agents at our disposal. In order to maintain any privacy of votes, we require at least one honest bit generator, at least $\Lambda = K - J + 1$ honest tellers, and a set of honest voters which contains at least two opposing voters. Without some opposition among the honest voters, the dishonest voters could subtract their sub-tally from the election tally and see that all honest voters cast the same votes. The remaining agents may all conspire, or some may just run their usual assigned protocols.

We will see that whenever the conspirators are able to gain better than chance information about the votes of honest voters, the conspirators may themselves be used to give information about residues.

Assume, without loss of generality, that the honest tellers are denoted by $T_1, T_2, \ldots, T_\Lambda$ and that bit generator $G_1$ is honest. Consider elections in which each teller $T_\lambda$ has selected parameters $(n_\lambda, y_\lambda)$. We will see that for any such parameter sets on which the conspirators have an advantage at distinguishing between votes, the conspirators may be used to, with a comparable advantage, decide residuosity modulo $n_\lambda$ for *at least* one $\lambda$, $1 \leq \lambda \leq \Lambda$. Intuitively, with this minimal set of honest election functionaries, an election is only as strong as the *weakest* of the $n_\lambda$ chosen. It will then be seen that obtaining even an inverse polynomial advantage at distinguishing between votes in an inverse polynomial fraction of all elections is sufficient to violate the prime residuosity assumption.

The reduction will show that if we are given a set of such $n_\lambda$ for which we do *not* know the factorizations, we can decide residuosity modulo at least one of the $n_\lambda$ with the same advantage as that with which the conspirators can distinguish between votes.

To accomplish this, we start by simulating an election with the given $n_\lambda$ as parameters. It will be shown that it is possible, by repeatedly starting, stopping, and restarting an election, to simulate an election with the same distribution that honest elections would have if the factors of the $n_\lambda$ were known. By assumption, therefore, the conspirators will demonstrate some advantage at distinguishing between votes in this case.

We next repeat the process except with $y_1$ changed from a random non-

residue (as specified by the protocol) to a random residue modulo $n_1$. We continue changing the $y_\lambda$, one at a time, from non-residues to residues until, for all $\lambda$ such that $1 \leq \lambda \leq \Lambda$, the $y_\lambda$ are $r^{\text{th}}$ residues modulo $n_\lambda$.

It will be shown that when all $y_\lambda$ are residues modulo their respective $n_\lambda$, a perfect symmetry exists, and that the conspirators will be completely unable to gain an advantage at distinguishing between votes since every vote could equally well represent a yes or a no. Since this is so, the initial $\varepsilon$ advantage of the adversaries must have dropped by at least $\varepsilon/\Lambda$ on some flip of a $y_\lambda$ from non-residue to residue. For this $n_\lambda$, we are able to decide residuosity by substituting unknown quantities for $y_\lambda$ and measuring the advantage attained.

### 5.10.2 Simulating Elections

The first task of the reduction is to show how to simulate elections, i.e. to simulate the duties of honest tellers when, in fact, the factors of their moduli are unknown. This puts us as the simulators in the somewhat odd situation of controling the "honest" agents and perhaps causing them to stray from their proper protocols while the "dishonest" agents are out of our control and may very well run proper protocols.

To begin with, each "honest" teller $T_\lambda$ posts its parameters $(n_\lambda, y_\lambda)$ and submits itself to challenges from inspectors which it must answer correctly. The "honest" inspectors are not a problem since they may transmit the correct challenge responses directly to the teller.

To answer the challenges of the "dishonest" inspectors, we must employ a technique which is common in proofs of interactive protocols. We observe that even probabilistic machines may be viewed as deterministic machines with access to random stimuli (usually a Turing machine with a random tape). If the machine is given the same stimuli, it will act in the same way. Thus, it is possible to run an adversary to a certain point multiple times — each time leaving the adversary in the same state. By altering later external inputs, it is possible to observe an adversary's responses to a variety of inputs at this point of the protocol.

To answer the challenges put forth by dishonest inspectors, a teller need only find, for each challenge $w_i$, one value $w_{i,j}$ for which the inspector will decompose *both* $w_{i,j}$ and $w_{i,j}w_i^{-1}$. (Recall that the interactive proof associated

with the challenges requires the inspector to decompose one or the other of $w_{i,j}$ and $w_{i,j}w_i^{-1}$ and that the teller decides which of the two is to be decomposed.) With decompositions of both $w_{i,j}$ and $w_{i,j}w_i^{-1}$, a teller can easily compute a decomposition of $w_i$ and thereby successfully answer the challenge. Thus, by stopping and restarting the adversary, each teller can answer the challenges to its parameters from any and all inspectors — even though it can not factor its own parameters.

When the "dishonest" voters cast their votes, each "honest" teller must determine its component of the vote cast. Without this information, the teller would not be able to complete the election by producing its tally-share in the final phase. In order to accomplish this a similar trick is used, but there is a catch here. The bits which are generated for the voter this time are not produced by the teller in question but rather by the bit generators. (If this were not so, others would have no confidence in the legitimacy of the vote.) This is where the one "honest" bit generator $G_1$ becomes involved. Since we, the election simulators, control $G_1$, we can change the value produced by $G_1$ at various times in order to change some of the bits to which voters must respond.

By changing the value of $G_1$'s bit, we are able to force each dishonest voter $V_\ell$ to reveal a decomposition of the vote which it cast. With these decompositions in hand, each teller $T_\lambda$ is easily able to complete the election simulation.

Note that after each stopping and restarting operation, the election state is restored and the election is then continued from that point. Thus, if all simulations were successful, the distribution of simulated elections would be identical to that of actual elections. Unfortunately, it may not *always* be possible to simulate an election.

**Definition** Formally, we say that an algorithm $\mathcal{A}$ is an $(r, N)$ *election simulator* with $1 - 1/P(N)$ success if for every parameter set

$$\langle (n_1, y_1), (n_2, y_2), \ldots, (n_\Lambda, y_\Lambda) \rangle$$

such that for each $\lambda$, $(r, n_\lambda, y_\lambda)$ is exact consonant and each $|n_\lambda| = N$, $\mathcal{A}$ can (without access to the factorization of one of the $n_\lambda$) run $T_1, T_2, \ldots, T_\Lambda$, $G_1$, and a set $H$ of honest voters "properly" in the sense that the actions produced by $\mathcal{A}$ are identical to those that would be produced by $T_1, T_2, \ldots, T_\Lambda$, $G_1$ and $H$ given that each $T_\lambda$ selected $(n_\lambda, y_\lambda)$ as its parameters. (Note that this is true even though $\mathcal{A}$ does not possess the factorization of one of the $n_\lambda$.)

In this definition of an election simulator, it is assumed that the simulator is given the factorizations of all but one of the parameters $n_\lambda$ for the tellers $T_\lambda$ which it must simulate. Although it is possible to simulate elections in which the factors of $n_\lambda$ are not known for *any* $\lambda$, it will be seen that for the reduction it is sufficient to simulate elections in which the factors of all but one $n_\lambda$ are known.

**Lemma 5.9** *For every prime $r$ and every polynomial $P(N)$, there exists an integer $N_0$ such that for all integers $N \geq N_0$ there exists a polynomial time $(r, N)$ election simulator $\mathcal{A}$ with $1 - 1/P(N)$ success.*

*Proof:*

Let $P(N)$ be an arbitrary polynomial. We give a constructive proof which defines an election simulator $\mathcal{A}$ which runs tellers $T_1, T_2, \ldots, T_\Lambda$, bit generator $G_1$, and the voters in $H$ properly in polynomial time with probability greater than $1 - 1/P(N)$.

Since we are only assuming control of the "honest" tellers $T_1, T_2, \ldots, T_\Lambda$, the "honest" bit generator $G_1$, and the "honest" voters $H$, we may assume that all other agents are "dishonest" and part of the conspiracy $\mathcal{C}$.

$\mathcal{A}$ begins by instructing each honest teller $T_\lambda$ to announce $(n_\lambda, y_\lambda)$ as its election parameters. All tellers $T_\lambda$ for which $\mathcal{A}$ possesses the factors of $n_\lambda$ can be simulated by $\mathcal{A}$ by executing the teller protocol precisely from this point on.

Denote the one teller $T_\lambda$ for which the factorization of $n_\lambda$ is not available as $T_*$ and denote its parameters by $(n_*, y_*)$. Simulating the actions of $T_*$ will require a significantly greater effort.

First, $\mathcal{A}$ must have $T_*$ answer each challenge satisfactorily. For each challenge, an inspector will prepare for teller $T_*$ a challenge value (a $w$ which is in either $\mathcal{RC}[0]_{(r,n_*,y_*)}$ or $\mathcal{RC}[1]_{(r,n_*,y_*)}$ for which the $T_*$ is supposed to be able to tell which) and a set of $\eta_1$ capsules each consisting of one element from each of $\mathcal{RC}[0]_{(r,n_*,y_*)}$ and $\mathcal{RC}[1]_{(r,n_*,y_*)}$.

Once all of the inspectors have prepared and posted all of their challenge capsules, teller $T_*$ is instructed by $\mathcal{A}$ to randomly select a subset of each of these sets of capsules.

Inspectors must then begin decomposing their subsets. Each subset must be decomposed by the inspector and one element of each remaining capsule must be "matched" with the corresponding $w_i$. (The match is accomplished by taking

an element $w_{i,j}$ from each associated capsule and decomposing $w_{i,j}w_i^{-1} \bmod n$.) Consider one such subset of capsules to be opened. If the inspector does not give these decompositions, then the challenge is invalid and should be ignored by $T_*$. If the inspector does present $T_*$ with appropriate decompositions, then $\mathcal{A}$ "backs up" the simulation and continues again from the point at which $T_*$ selected the subset of these capsules. Here, $\mathcal{A}$ instructs $T_*$ to select another random subset of the capsules. The simulation is continued until these capsules are dealt with. There are two possibilities here. The inspector may, once again, present $T_*$ with appropriate decompositions. In this case, if the subset of capsules is different, there must be at least one capsule which the inspector both decomposed and matched with $w_i$. This is sufficient for $\mathcal{A}$ to determine $[\![w_i]\!]$ and thereby instruct $T_*$ as to the proper answer to the challenge. If the inspector this time refuses to present $T_*$ with appropriate decompositions, then $\mathcal{A}$ once again backs up the simulation and instructs $T_*$ to try another subset. Let $\xi_1 = 2I\eta_2 P(N)$. This process is repeated up to $\xi_1$ times or until $\mathcal{A}$ is able to get each inspector to present a second set of decompositions for each challenge and thereby allow $\mathcal{A}$ to compute each $[\![w_i]\!]$. If after $\xi_1$ attempts, $\mathcal{A}$ is unable to get an inspector to present a second set of decompositions for a given challenge, then the simulation fails. If, however, the appropriate decompositions are obtained, then the simulation can continue.

Once the challenge phase of the election is completed, the ballot preparation phase can begin. $\mathcal{A}$ instructs each "honest" voter in $H$ to execute its normal protocol. $\mathcal{A}$ must then prepare to deal with the "dishonest" voters.

Each dishonest voter prepares a master ballot $B$ and $\eta_3$ additional "scratch" ballots for each bit generator. Once all ballots are prepared, $\mathcal{A}$ instructs bit generator $G_1$ to select a random subset of the scratch ballots associated with it. (Other bit generators will also select subsets of the scratch ballots associated with them). Voters must then decompose their ballots. Each voter decomposes the designated ballots (those in the selected subset) and for each remaining ballot $B'$ must decompose either $B'/B$ or $B'\backslash B$. If the voter does not present these decompositions, it is acting improperly and should be ignored. Once the decompositions are obtained, $\mathcal{A}$ backs up the simulation and instructs $G_1$ to select another random subset of the ballots associated with it. For each voter, if this second subset is decomposed appropriately, then (if the subset is different), there must be some ballot $B'$ for which the voter has presented

decompositions of both $B'$ and either $B'/B$ or $B'\backslash B$. This is sufficient for $\mathcal{A}$ to completely decompose $B$. Let $\xi_2 = 2LP(N)$. This process is repeated up to $\xi_2$ times or until $\mathcal{A}$ is able to get every voter to present a second set of decompositions and thereby allow $\mathcal{A}$ to compute a decomposition of $B$. If $\mathcal{A}$ is unable to obtain a second set of decompositions for some voter, the simulation fails. If, however, a decomposition of $B$ is obtained for all voters, then the simulation can continue.

In the voting phase, $\mathcal{A}$ simply instructs the set $H$ of honest voters to vote according to either assignment $h_0$ or $h_1$ depending upon the desired simulation.

Since the evaluation function $\Psi$ is a homomorphism, the $(\oplus)$ sum of the decompositions of the votes cast is a decomposition of the $(\circ)$ product of the votes cast. Thus, if there has been no failure up to this point, $\mathcal{A}$ has a complete decomposition of the vote product and can complete the election directly. $\mathcal{A}$ simply instructs each honest teller $T_\lambda$ to release as its tally-share the class of $\lambda^{\text{th}}$ component of the vote product as given by the decomposition. Since $\mathcal{A}$ has a complete decomposition, the interactive proof of these tally-shares can be completed normally.

It is clear that with the choices of $\xi_1$ and $\xi_2$ given, this simulation process is polynomial in $K$, $L$, $M$, and $N$.

We must now analyze the probability of a successful simulation. The simulation succeeds unless one of the inspectors or voters gives a proper first set of decompositions but fails to give a second set of decompositions on any of the subsequent tries. Since these cases are indistinguishable to the conspirators, we may assume that all trials are independent.

It is the "goal" of the conspirators to keep $\mathcal{A}$ from simulating an election successfully. Therefore, we may assume that the conspirators will try to maximize the probability of causing at least one inspector or voter to decompose the first set of capsules/ballots and to decompose no further sets. This probability is, of course, maximized by maximizing the probability of each inspector and each voter of achieving this goal.

Let $p_I$ be the probability that an inspector will decompose a given subset of capsules (remember that $p_I$ is independent of the iteration, since each iteration looks like the first, and that $p_I$ is independent of the inspector, since each is trying to maximize the probability that it will cause the simulation to fail). The probability that a given inspector will, on a given challenge, decompose

the first subset and none of the $\xi_1$ subsequent subsets is

$$p_I(1 - p_I + 2^{-\eta_1})^{\xi_1}.$$

(The $2^{-\eta_1}$ term accounts for the possibility that the same subset is chosen since this will not allow the challenge to be answered.) Each of the $I$ inspectors can challenge the honest teller $T_*$ up to $\eta_2$ times. Thus, the total number of challenge responses which must be successfully simulated is bounded by $I\eta_2$.

Similarly, let $p_V$ be the probability that a voter will decompose a given subset of ballots. The probability that a given voter will decompose the first subset and none of the $\xi_2$ subsequent subsets is

$$p_V(1 - p_V + 2^{-\eta_3})^{\xi_2}.$$

There are $L$ eligible voters, so the total number of ballots which must be successfully decomposed is bounded by $L$.

Now, to see where the function $f(p) = p(1-p+2^{-\eta})^{\xi}$ reaches its maximum value, we may differentiate:

$$f'(p) = (1 - p + 2^{-\eta})^{\xi} - \xi p(1 - p + 2^{-\eta})^{\xi-1}.$$

Therefore $f'(p) = 0$ when

$$p = 1 + 2^{-\eta} \quad \text{or} \quad p = \frac{1 + 2^{-\eta}}{\xi + 1}.$$

Clearly, the maximum is achieved at $p = (1+2^{-\eta})/(\xi+1)$. Thus, the maximum value attained by the function $f(p)$ is

$$\begin{aligned} f\left(\frac{1 + 2^{-\eta}}{\xi + 1}\right) &= \left(1 + 2^{-\eta}\right)^{\xi+1} \frac{\xi^{\xi}}{(\xi+1)^{\xi+1}} \\ &= \left(1 + 2^{-\eta}\right)^{\xi+1} \frac{1}{\xi}\left(\frac{\xi}{\xi+1}\right)^{\xi+1} \\ &= \frac{1}{\xi}\left(1 + 2^{-\eta}\right)^{\xi+1}\left(1 - \frac{1}{\xi+1}\right)^{\xi+1} \\ &< \frac{1}{\xi}\left(e^{2^{-\eta}(\xi+1)}e^{-1}\right), \end{aligned}$$

where $e$ is Euler's constant. In particular, when $2^{\eta} \geq \xi + 1$, then $f(p) < \frac{1}{\xi}$.

Thus, with

$$\eta_1 = \lceil (\log_2 I)(\log_2 \log_2 K)N \rceil$$

and

$$\xi_1 = 2I\eta_2 P(N) = 2I\lceil (\log_2 K)N \rceil P(N),$$

$2^{\eta_1} \geq \xi_1$ for sufficiently large $N$. Also, with $\eta_3 = \lceil (\log_2 L)N \rceil$ and $\xi_2 = 2LP(N)$, $2^{\eta_3} \geq \xi_2$ for sufficiently large $N$.

Thus, the probability of at least one inspector challenge causing the simulation to fail is less than

$$\frac{I\eta_2}{\xi_1} = \frac{1}{2P(N)}$$

and the probability of at least one voter causing the simulation to fail is less than

$$\frac{L}{\xi_2} = \frac{1}{2P(N)}.$$

Therefore, the probability of a successful simulation is greater than

$$1 - 1/P(N)$$

for sufficiently large $N$. ∎

**Remark** Such a simulation could be accomplished even if factorizations of *none* of the $n_\lambda$ were available. This would simply require increasing $\eta_1$ by a factor of $\log K$ and $\xi_1$ by a factor of $K$ to account for the additional challenges which would have to be answered by the back up and repeat process.

### 5.10.3 The Symmetry Condition

It is not at all hard to see that in "pseudo-elections" in which the "honest" tellers $T_1, T_2, \ldots, T_\Lambda$ ($\Lambda = K - J + 1$) select parameters $(n_\lambda, y_\lambda)$, $1 \leq \lambda \leq \Lambda$, where each $y_\lambda$ *is* an $r^{\text{th}}$ residue modulo $n_\lambda$, distinct vote assignments are indistinguishable in an information theoretic sense. This is so because *every* vote assignment with the proper sub-tally among the honest voters is completely consistent with such a pseudo-election!

When $y \in \mathbf{Z}_\mathbf{n}^\mathbf{r}$, then all of the residue classes $\mathcal{RC}[c]$ are identical. Thus, if $x$ is chosen randomly from $\mathbf{Z}_\mathbf{n}^*$, then for *every* integer $c$, $w = y^c x^r \bmod n$ represents a uniform selection from among the elements of $\mathbf{Z}_\mathbf{n}^\mathbf{r}$. Thus, each $w$

could equiprobably be used to indicate any class. Hence, any vote could just as easily represent any other, and all consistent vote assignments become equally meaningful. Therefore in this case, *no* possible adversary could do better than chance at distinguishinging between vote assignments.

**Definition** A $((n_1, y_1), (n_2, y_2), \ldots, (n_\Lambda, y_\Lambda))$ pseudo-election is a transcript of an election for which *check* = good and for each teller honest teller $T_\lambda$ has selected parameters $(n_\lambda, y_\lambda)$.

**Lemma 5.10** *Let $H$ be a set of honest voters and let $h_0$ and $h_1$ be two assignments of votes to voters in $H$ which have the same sub-tally over $H$. Let $((n_1, y_1), (n_2, y_2), \ldots, (n_\Lambda, y_\Lambda))$ be such that each $y_\lambda$ is in $\mathbf{Z}^{\mathbf{r}}_{\mathbf{n}_\lambda}$. Every pseudo-election transcript in which the voters in $H$ vote according to $h_0$ could also be obtained by the voters in $H$ voting according to $h_1$, and vice-versa.*

*Proof:*

A vector $C = \langle c_1, c_2, \ldots, c_K \rangle$ such that $\|C\|_J = t$ can be uniformly selected by selecting $J-1$ random values uniformly from $\mathbf{Z_r}$ and by filling the remaining $\Lambda = K - (J-1)$ values with the unique elements of $\mathbf{Z_r}$ such that $\|C\|_J = t$ is true. Thus, in a $J$-threshold $K$-teller $L$-voter election, a vote of type $t$ can be generated uniformly by first selecting $J-1$ random values uniformly from $\mathbf{Z_r}$ and $\Lambda$ additional values to form a vector $C$ with $\|C\|_J = t$ and then uniformly selecting a vote $W$ with disclosure $C$.

Thus, being given $J-1$ components of the disclosure of a uniformly generated vote of any type $t$, gives no information as to the type $t$. If each $y_\lambda \in \mathbf{Z}^{\mathbf{r}}_{\mathbf{n}_\lambda}$, then for each $(n_\lambda, y_\lambda)$, the residue classes $\mathcal{RC}[c]_{(r, n_\lambda, y_\lambda)}$ coincide for all integers $c$. Thus, if a "vote" is formed by uniform selection with election parameters which include these $(n_\lambda, y_\lambda)$, then the $\Lambda$ components corresponding to these parameters will simply consist of random $r^{\text{th}}$ residues. Since the remaining $J-1$ components, although they have well-defined disclosures, are selected completely uniformly and independently of the intended type, a uniformly generated "pseudo-vote" $W$ will be perfectly consistent with any intended type.

Thus, all "votes" in such an election are completely consistent with all types, and pseudo-elections in which the honest voters in $H$ vote according to assignment $h_0$ are identical to pseudo-elections in which the honest voters in $H$ vote according to assignment $h_1$. ∎

### 5.10.4 The Formal Reduction

**Definition** A vector $\nu = \langle n_1, n_2, \ldots, n_\Lambda \rangle$ is said to be $(r, N)$-*legitimate* if for every $n_\lambda \in \nu$, $|n_\lambda| = N$ and the pair $(r, n_\lambda)$ is exact consonant.

**Definition** An election is said to be $(r, N, \nu)$-honest if $\nu = \langle n_1, n_2, \ldots, n_\Lambda \rangle$ is $(r, N)$-legitimate and if there exists a set $T_1, T_2, \ldots, T_\Lambda$ of honest tellers such that each $n_\lambda$ is the first component of the election parameters selected by the teller $T_\lambda$.

The object now is to see how to take a specific given $n$ and gain an advantage at deciding residuosity modulo this $n$. For many such $n$, no advantage will be attained; but if it is possible to attain an inverse polynomial advantage at compromising elections with security parameter $N$, then, when $N$ is sufficiently large, an inverse polynomial advantage will be obtained at deciding residuosity modulo $n$ for a polynomially sized fraction of the possible values of $n$ of size $N$.

Recall from section 5.3 that a conspiracy $C$ is said to compromise privacy if there exists some polynomial $P(N)$ such that for infinitely many values of $N$, $C$ can gain a $1/P(N)$ advantage at distinguishing between some pair of vote assignments in random elections with security parameter $N$ in time $P(N)$.

**Theorem 5.11** *The prime residuosity assumption implies that there exists no conspiracy $C$ that compromises privacy.*

*Proof:*

We prove the theorem by contraposition.

Fix $r$ to be a prime and assume that there exists some conspiracy $C$ that compromises privacy. By assumption, $C$ is able to gain a $1/P(N)$ advantage at distinguishing between some pair vote assignments in $(r, N, \nu)$-honest elections for infinitely many $N$.

Consider first the "randomizing function" $f_n(y)$ which is defined to select an integer $c$ uniformly with $0 < c < r$ and an $x$ uniformly from $\mathbf{Z}_\mathbf{n}^*$ and yield $f_n(y) = y^c x^r \bmod n$. This randomizing function has the interesting property that when $(r, n)$ is exact consonant then if $y \in \mathbf{Z}_\mathbf{n}^\mathbf{r}$, $f_n(y)$ is a uniformly selected member of $\mathbf{Z}_\mathbf{n}^\mathbf{r}$, and if $y \in \overline{\mathbf{Z}_\mathbf{n}^\mathbf{r}}$, then $f_n(y)$ is a uniformly selected member of $\overline{\mathbf{Z}_\mathbf{n}^\mathbf{r}}$.

We now define the family of algorithms $\mathcal{B}_i$. For $1 \leq i \leq \Lambda$, let $\mathcal{B}_i$ be an algorithm which takes as input the pair $(n, y)$ and acts as follows. $\mathcal{B}_i$ uniformly selects for $1 \leq \lambda \leq \Lambda$, $\lambda \neq i$, values $n_\lambda$ such that each $|n_\lambda| = N$ and each pair $(r, n_\lambda)$ is exact consonant (as in lemma 2.27). Then, for $1 \leq \lambda < i$, $\mathcal{B}_i$

uniformly selects $y_\lambda \in \mathbf{Z}^{\mathbf{r}}_{\mathbf{n}_\lambda}$, and for $i < \lambda \leq \Lambda$, $\mathcal{B}_i$ uniformly selects $y_\lambda \in \overline{\mathbf{Z}^{\mathbf{r}}_{\mathbf{n}_\lambda}}$. (Note that $\mathcal{B}_i$ can retain the factorizations of these $n_\lambda$, so generating these $y_\lambda$ as prescribed is not a problem.) $\mathcal{B}_i$ also selects a random bit $b \in \{0,1\}$. $\mathcal{B}_i$ then sets $n_i = n$ and $y_i = f_n(y)$ and (by lemma 5.9) simulates an election with parameters $\langle (n_1, y_1), (n_2, y_2), \ldots, (n_\Lambda, y_\Lambda) \rangle$ and with an assignment $h_b$ of votes to honest voters in $H$ where $h_0$ and $h_1$ are assignments of votes to voters in $H$ which $\mathcal{C}$ is assumed to distinguish between with an inverse polynomial advantage. Unless the simulation fails, the output of $\mathcal{C}$ is a bit $c \in \{0,1\}$. If the simulation fails, we set $c = 0$. The output of $\mathcal{B}_i$ is the exclusive-or $b \oplus c$. Thus, (except in the case where the simulation fails) output$(\mathcal{B}_i) = 0$ precisely when output$(\mathcal{C})$ matches the choice of the vote assignment $h_b$ on which $\mathcal{C}$ is run.

For the most part, there is no guarantee that $\mathcal{C}$ will do anything at all with these simulated elections (i.e. the output of $\mathcal{C}$ could always be 0) since a powerful $\mathcal{C}$ may be able to detect that some of the $y_\lambda$ are $r^{\text{th}}$ residues and might therefore refuse to participate in an election or cause the simulation to always fail. There are, however, some constraints that can be placed on $\mathcal{C}$. Let $p_0$ be the probability that $\mathcal{C}$ outputs 1 given that $b = 0$. Let $p_1$ be the probability that $\mathcal{C}$ outputs 1 given that $b = 1$. Let $q_i$ denote the probability that output$(\mathcal{B}_i) = 1$,

First of all, as $n$ varies among values such that $(r, n)$ is exact consonant and $y \in \overline{\mathbf{Z}^{\mathbf{r}}_{\mathbf{n}}}$, then the simulated elections represent a uniform sampling of actual elections with $N = |n|$. By assumption, $\mathcal{C}$ gains a $1/P(N)$ advantage at distinguishing between $h_0$ and $h_1$ for infinitely many $N$. By definition, this implies that $|p_1 - p_0| > 1/P(N)$. By lemma 5.9, we may run the simulator $\mathcal{B}_1$ such that it succeeds with probability greater than $1 - 1/(2P(N))$. Therefore, when $y$ is not an $r^{\text{th}}$ residue, $|q_1 - 1/2| > 1/(2P(N))$.

Next, when $B_\Lambda$ is given an $n$ such that the pair $(r, n)$ is exact consonant and a $y \in \mathbf{Z}^{\mathbf{r}}_{\mathbf{n}}$, then lemma 5.10 applies and hence $\mathcal{C}$ can attain no advantage whatsoever at distinguishing between $h_0$ and $h_1$. Thus, in this case, $q_\Lambda = 1/2$.

Finally, as $n$ varies among values such that $(r, n)$ is exact consonant, $\mathcal{B}_i$, when given a $y \in \mathbf{Z}^{\mathbf{r}}_{\mathbf{n}}$, will be indistinguishable from $\mathcal{B}_{i+1}$, when given a $y \in \overline{\mathbf{Z}^{\mathbf{r}}_{\mathbf{n}}}$. Thus, the same advantage at distinguishing between $h_0$ and $h_1$ will be obtained in both of these cases.

For a fixed $N$, let $a_i$ denote the overall value $|q_i - 1/2|$ when the input $(n, y)$ to $\mathcal{B}_i$ is such that $(r, n)$ is exact consonant, $|n| = N$, and $y \in \overline{\mathbf{Z}^{\mathbf{r}}_{\mathbf{n}}}$; and let

$b_i$ denote the overall value $|q_i - 1/2|$ when the input $(n, y)$ to $\mathcal{B}_i$ is such that $(r, n)$ is exact consonant, $|n| = N$, and $y \in \mathbf{Z}_n^r$. The above arguments give the following chain.

$$1/(2P(N)) = a_1, b_1 = a_2, \ldots, b_{\Lambda-1} = a_\Lambda, b_\Lambda = 0$$

This implies that, for each of the infinitely many $N$ for which $C$ gains a $1/P(N)$ advantage at distinguishing between $h_0$ and $h_1$, there exists some $i$ such that $|a_i - b_i| \geq 1/(2\Lambda P(N))$. Let $\iota$ denote a value for which $|a_\iota - b_\iota| \geq 1/(2\Lambda P(N))$ for infinitely many values of $N$. (There must exist at least one such $\iota$.)

We now have that for infinitely many values of $N$, the algorithm $\mathcal{B}_\iota$ when given inputs $n$ with $|n| = N$ and $y$ with $y \in \overline{\mathbf{Z}_n^r}$ has a probability of outputting 1 which differs by at least $1/(2\Lambda P(N))$ from the probability of outputting 1 when $|n| = N$ and $y \in \mathbf{Z}_n^r$. This $\mathcal{B}_\iota$ will be used to develop an algorithm which decides $r^{\text{th}}$ residues.

Let $\mathcal{A}$ be an algorithm which takes as input integers $n$ and $y$ with $y \in \mathbf{Z}_n^*$. $\mathcal{A}$ runs $\mathcal{B}_\iota$ on the pair $(n, y)$. For infinitely many values of $N$, $\mathcal{B}_\iota$ distinguishes between the case $y \in \mathbf{Z}_n^r$ and the case $y \in \overline{\mathbf{Z}_n^r}$ with a $1/(2\Lambda P(N))$ advantage when $|n| = N$ and $(r, n)$ is an exact consonant pair.

For convenience, we say that a given $n$ is *decided* if for that $n$, $\mathcal{B}_\iota$ gains more than a $1/(4\Lambda P(N))$ advantage at distinguishing between residues and non-residues. We will see that at least a $1/(4\Lambda P(N))$ fraction of the $n$ of size $N$ such that $(r, n)$ is exact consonant are decided by $\mathcal{B}_\iota$.

The worst case occurs when, for as many $n$ as possible, $\mathcal{B}_\iota$ gains exactly a $1/(4\Lambda P(N))$ advantage. The overall advantage, however, must be $1/(2\Lambda P(N))$. To further the worst-case scenario, assume that each $n$ which *is* decided is decided with certainty (i.e. with advantage 1). If only a $1/(4\Lambda P(N))$ fraction of the $n$ were decided, it would still not raise the overall advantage to quite $1/(2\Lambda P(N))$. Thus, at least a $1/(4\Lambda P(N))$ fraction of the $n$ of size $N$ such that $(r, n)$ is exact consonant are decided by $\mathcal{B}_\iota$.

Hence, for infinitely many $N$, $\mathcal{A}$ decides $r^{\text{th}}$ residues with a $1/(4\Lambda P(N))$ advantage for at least a $1/(4\Lambda P(N))$ fraction of the $n$ of size $N$ such that the pair $(r, n)$ is exact consonant, and this advantage is achieved in time polynomial in $N$. This violates the prime residuosity assumption, and therefore the theorem is proven. ∎

## 5.11 Usage and Optimizations

For a relatively large election — perhaps $L = 100,000,000$ voters, $K = 100$ tellers, $M = 100$ bit generators, $I = 100$ inspectors, and confidence $1 - 2^{-100}$ — the computations required by the election schema given here are near the edge of feasibility on powerful machines. There are several approaches to making the entire process more managable.

### 5.11.1 Parallelization

First, the entire schema is massively parallelizable. All of the repeated steps in every sub-phase can be performed in parallel, and with a large number of processors, each voter protocol can be performed with only $\mathcal{O}(N)$ sequential $N$-bit multiplications and divisions. The time to complete the teller protocols is dominated by the time to determine residue classes. Besides being able to handle separate inspector challenges in parallel, the compilation of a residue class table of $r^{\text{th}}$ roots of unity or the search for a recognized $r^{\text{th}}$ root of unity can easily be parallelized. Finally, the *check* function can also easily be run with very few sequential dependencies.

### 5.11.2 Asynchrony

Although the definitions of an election system given in section 5.3 require that the processors be synchronous, it does not seem that this is necessary for the election schema given in section 5.6. Within each of the four election phases, there are as many as six major steps to be performed. The processors need not run synchronously as long as all processors have completed each step before beginning the next.

### 5.11.3 Other Streamlining

Even when run sequentially, there are several variants which may be used to make the election schema more efficient. First, if an unpredictable random source can be found and agreed upon, then the set $\mathcal{G}$ of bit generators can be reduced to size $M = 1$. As mentioned earlier, a bit generator of this form is often called a *beacon* ([Rabi83a]).

Even when no beacon is available, the bit generators may use the exclusive-or of their individual bits to form a simulated beacon. In order to accomplish this, the bit generators select parameters and answer challenges from inspectors just as do the tellers. Each generator then can use its parameters to encrypt each bit as it is released. When all bit generators have released an encrypted bit, the decryptions are revealed and the exclusive-or of the decrypted bits is used as a simulated beacon bit. This prevents the last generator from deciding the outcome of the simulated beacon bit.

To take this approach a step further, the tellers can *be* the bit generators. Tellers can use their (already chosen and tested) parameters to encrypt bits to simulate a beacon. This combination is quite appealing since the security of the election already assumes that at least $K - J + 1$ tellers are honest, and only one honest bit generator is needed to ensure correctness.

The choice of the inspector set $I$ can play a large role in the efficiency of an election. The inspector set can consist of the entire electorate (all voters), but this can be very expensive. Instead of each voter challenging each teller, each bit generator can challenge each teller. (In the case where the tellers and the bit generators are the same set, each teller tests every other.) Since it is necessary to trust at least one bit generator anyway, the bit generators can act as surrogates for the purpose of testing the tellers. This does not work, however, if the bit generators are natural sources (or a beacon) rather than generators, since a challenge requires computation and the maintainance of secret information. The tellers themselves can, of course, act as the inspectors and challenge each other. This makes good sense if the tellers are already acting as the bit generators, as mentioned previously.

## 5.12 Extensions and Variations

The secret-ballot election schema described in this chapter is quite flexible and adaptable to a variety of purposes and changing circumstances. Some of these variations will be described below.

### 5.12.1 Multiway Elections

So far, we have considered only elections with two possible choices—"yes" and "no". The schema presented extends easily to elections where many choices are allowed.

One method to hold a three-way election, for example, would be to select an integer $\tilde{r}$ greater than the number of eligible voters and to choose the election prime $r$ to be greater than $\tilde{r}^2$ (rather than just greater than the number of eligible voters). Votes cast by voters would be constrained to one of the following three types: choice (A) – type $\tilde{r}$; choice (B) – type 1; and choice (C) – type 0. Three component capsules are used to constrain the votes to be of one of these three forms. Since $\tilde{r}$ is greater than the number of eligible voters and $r > \tilde{r}^2$, the product of the votes will be of a type $t$ which is unique modulo $r$, and this $t$ in turn is uniquely expressible as $t = a\tilde{r} + b$ for $0 \leq b < \tilde{r}$. The values of $a$ and $b$ respectively denote the number of votes for choices (A) and (B), and the remaining valid votes are for choice (C). It is easy to see how to extend this approach to elections with more than three choices.

Moti Yung has suggested an alternative method for holding multiway elections. The approach is somewhat similar to the Boolean circuit satisfiability scheme described in section 3.3.5. For a three-way election, three component capsules are again used. Each component of the (unordered) three component capsules would consist of an *ordered* triple of votes. The ordered triple would consist of one yes vote and two no votes. The yes would appear in each of the three ordered positions exactly once in each capsule. That is, a capsule would look like $\{(\text{no}, \text{no}, \text{yes}), (\text{no}, \text{yes}, \text{no}), (\text{yes}, \text{no}, \text{no})\}$, and the actual "vote" cast would be a triple of one of these three forms. Three separate counters are maintained — one corresponding to each of three choices (A), (B), and (C). A vote triple of the form $(\text{no}, \text{yes}, \text{no})$, for example, would put a no vote into the (A) counter, a yes vote into the (B) counter, and a no vote into the (C) counter. The product of the votes in each of the three counters is taken separately, and a tally of each of the three counters is formed as in the general schema. Each counter tally then represents the number of votes cast for the corresponding choice. Once again, it is easy to see how this method can be extended to elections with more than three choices.

### 5.12.2 Preferential Voting

The previous methods enable the production of tallies in elections with more than two choices, but they do not indicate how to make use of these tallies to decide an issue such as which of three candidates will win an elective office.

One may, of course, simply declare the candidate with the largest number of votes to be the winner. This so called *plurality* rule suffers from several serious problems. It is possible with plurality voting, for instance, to elect a candidate who has only minority support (and, in fact, strong majority disapproval).

Additional flexibility can be obtained by allowing each voter one of six choices. These choices would correspond to the six possible preference rankings among the three candidates. The number of voters who select each of the six preference lists could then be determined, and a preferential voting method could be applied to these results.

Unfortunately, the results of Kenneth Arrow ([Arro63]) indicate that, even with a complete list of voter preferences, no voting method exists which will always choose a winner which satisfies a small number of highly desirable criteria. Even though Arrow's work shows that no voting method is perfect, there are alternatives to plurality voting, and once the list of voter preferences has been compiled, any voting rule can be used to determine a winner.

One quite interesting rule is called *approval* voting (see [Stra80]). With approval voting, each voter can cast one or zero votes in favor of each candidate, and the candidate who receives the most votes is the winner. It can be shown that, with approval voting, it is in each voter's interest to cast votes for all candidates above a certain threshold of acceptability. Approval voting is particularly useful when there are several identical electoral positions to be filled (such as at-large seats on a city council). Here the candidates with the most votes fill the seats.

Approval voting can be implemented quite easily within the general secret-ballot election schema. Each voter prepares a ballot for each candidate (exactly as ballots were prepared before). Then each voter casts one vote from each ballot for each candidate. In this way, each candidate's tally can be computed.

### 5.12.3 Giving the Winner without the Tally

Adi Shamir suggested the problem of holding a secret-ballot election in which all participants are confident of the winner, but in which the actual tally of the election is not released. Shamir, together with Oded Goldreich and Ron Rivest, offered ideas which led to a solution limited to the single teller (centralized government) case.

The idea is to hold an election as usual except for the release of the tally. At this point, instead of showing that a specific tally is the case, the government shows that *all* possible tallies which would cause the losing choice to win *are not* the case. This can be accomplished by using the interactive proof method of section 3.3.1 up to $r/2$ times to show that the vote product is not a member of any of the up to $r/2$ possible residue classes which would cause the losing choice to win. This solution is described in somewhat greater detail in [Cohe86].

### 5.12.4 Related Schemas

In addition to the election schema presented in this chapter, a number of related schemas which all conform to the election paradigm of Figure 5.1 can be devised. One of these schemas is based upon the difficulty of computing the discrete logarithm modulo a known prime ([Adle79], [PoHe78], [COS86]). Another is based upon the difficulty of determining the order of an element modulo the product of two unknown primes. A third proposed schema encrypts a vote using the low order bit of a message encrypted by RSA [RSA78].

The major requirement of such a schema seems to be the existance of a function that can be computed on a set of encrypted data which preserves the sum of the unencrypted components (see [RAD78]). The variety of number theoretic problems on which such schemes can be based and the ease with which they have been found gives additional reason to believe in the usefulness of the general paradigm.

# Chapter 6

# Conclusions

It does not seem likely that the methods for holding a verifiable secret-ballot election presented in this thesis will be used in actual elections in the near future. The schema is reasonably practical, and advances in technology can be expected to make the schema quite fast; however, public acceptance and the wide-scale availability of the required technology seem to be some distance away.

The process of implementing a schema such as this is political and not technical, and in some regions it is not in the interests of those in power to have verifiable elections. There are many aspects of elections which are simply beyond the scope of this work. The maintainance of eligible voter rolls, for instance, is not addressed at all. In fact, all *public* aspects of elections are completely independent of this work. The results given in this thesis merely allow the private component of secret-ballot elections to be managed as though it, too, were public.

It is also true in our current society that distrust of science and scientists is widespread. It does not seem to be satisfactory to implement a "verifiable" secret-ballot election schema that will only serve to convince a select few who understand the mathematics. Fortunately, the mathematics in this schema is, although somewhat detailed, accessible at the undergraduate level or even conceivably the high school level. If an election schema such as this were to be adopted for large-scale use, it could be accompanied by an effort to educate people about the mechanism involved. A short undergraduate mathematics course or even an advanced high school course should be sufficient for this

purpose.

One of the motivations for this work has been the desirability for preferential elections. The current "plurality" voting rules most frequently used seem all too often to place a candidate with a minority mandate into office. The addition of run-off elections which are used in some areas improve the prospects slightly, but they still leave much to be desired.

Although Kenneth Arrow has shown that no voting rule can be completely satisfactory, the implementation of an (albeit imperfect) preferential voting system could improve current systems tremendously. It seems, however, that large-scale preferential voting is beyond our ability to implement with paper or mechanical voting technology. Electronic technologies could, however, be used for this purpose.

In fact, it is ever more the case that current elections are begin conducted with electronic technologies. This is especially true for the process of vote counting.

As the electronic component to elections becomes more pronounced, our confidence in both the privacy of our votes and the accuracy of the tally should wane. It is generally felt that small-scale corruption or errors in manual or mechanical voting systems would lead to only a small number of erroneous votes or violated privacies. Such is not the case, however, with electronic systems where small "bugs" (intential or unintentional) can easily lead to huge shifts of votes or the public disclosure of large numbers of votes. It seems evident that if we are to bring computerization into our electoral processes, then we must do it in such a way as to preserve the integrity of the process and to prevent the concentration of power into the hands of the few who control the process. The adoption of a schema of the sort presented here could do much towards serving these ends.

It is possible to envision a society in which small voting devices or voting software are available from a variety of vendors. Such devices or software could perhaps be "plugged" into voting "outlets" at various centers or even in the home.

The advantages that could be attained by implementing such a system are numerous. They include the possibility of preferential voting (perhaps made simple through an interactive system), confidence in the outcome, confidence in privacy, greater ease and flexibility in voting, and far greater speed in com-

puting results. It would even be possible (although probably not effective) to implement a modern Athenian democracy in which major issues are brought before the public for a rapid vote.

Many of these ideas may seem overly ambitious and unrealistic, but we should be aware that the technology exists to make them possible. If the will of society were to shift towards these goals, the technology required to achieve them could be ready.

Of more immediate interest are the tools that have been built which allow for the construction of verifiable secret-ballot elections. The election encryption function gives a high-density method of probabilistic encryption which has a wide variety of applications. Cryptographic capsules have applications to current needs in user authentication and in electronic funds transfer, for example. Secret sharing homomorphisms have applications in many instances where computing of shared, encrypted data arises. And, even if it is somewhat wistful, an important application of all of these methods makes possible the holding of large-scale, robust, and verifiable secret-ballot elections.

# Bibliography


[AdHu87] **Adleman, L.** and **Huang, M.** "Recognizing Primes In Random Polynomial Time." *Proc.* 19th *ACM Symp. on Theory of Computing*, New York, NY (May 1987), 462–469.

[Adle79] **Adleman, L.** "Subexponential Algorithm for The Discrete Logarithm Problem." *Proc.* 20th *IEEE Symp. on Foundations of Computer Science*, San Juan, PR (Oct. 1979), 55–60.

[Adle80] **Adleman, L.** "On Distinguishing Prime Numbers from Composite Numbers." *Proc.* 21st *IEEE Symp. on Foundations of Computer Science*, Syracuse, NY (Oct. 1980), 387–406.

[AdMc82] **Adleman, L.** and **McDonnell, R.** "An Application of Higher Reciprocity to Computational Number Theory." *Proc.* 23rd *IEEE Symp. on Foundations of Computer Science*, Chicago, IL (Nov. 1982), 100–106.

[AFK87] **Abadi, M.**, **Feigenbaum, J.**, and **Kilian, J.** "On Hiding Information from an Oracle." *Proc.* 19th *ACM Symp. on Theory of Computing*, New York, NY (May 1987), 195–203.

[Angl82] **Angluin, D.** "Lecture Notes on the Complexity of Some Problems in Number Theory." *TR-243, Yale University, Department of Computer Science*, New Haven, CT (Aug. 1982).

[AnLi83] **Angluin, D.** and **Lichtenstein, D.** "Provable Security of Cryptosystems: a Survey." *TR-288, Yale University, Department of Computer Science*, New Haven, CT (Oct. 1983).

[APR83] **Adleman, L., Pomerance, C.**, and **Rumley, R.** "On Distinguishing Prime Numbers from Composite Numbers." *Annals of Math. 117*, (1983), 173–206.

[Arro63] **Arrow, K.** *Social Choice and Individual Values.* John Wiley and Sons, New York (1963).

[AsBl80] **Asmuth, C.** and **Bloom, J.** "A Modular Approach to Key Safeguarding." *Texas A&M University, Department of Mathematics*, College Station, TX (1980).

[Baba85] **Babai, L.** "Trading Group Theory for Randomness." *Proc.* 17th *ACM Symp. on Theory of Computing*, Providence, RI (May 1985), 421–429.

[Bena86a] **Benaloh, J.** "Cryptographic Capsules: A Disjunctive Primitive for Interactive Protocols." *Crypto '86*, Santa Barbara, CA (Aug. 1986).

[Bena86b] **Benaloh, J.** "Secret Sharing Homomorphisms: Keeping Shares of a Secret Secret." *Crypto '86*, Santa Barbara, CA (Aug. 1986).

[BenO81] **Ben-Or, M.** "Probabilistic Algorithms in Finite Fields." *Proc.* 22nd *IEEE Symp. on Foundations of Computer Science*, Nashville, TN (Oct. 1981), 394–398.

[BenO83] **Ben-Or, M.** "Another Advantage of Free Choice: Completely Asynchronous Agreement Protocols." *Proc.* 2nd *ACM Symp. on Principles of Distributed Computing*, Montreal, PQ (Aug. 1983), 27–30.

[BeYu86] **Benaloh, J.** and **Yung, M.** "Distributing the Power of a Government to Enhance the Privacy of Voters." *Proc.* 5th *ACM Symp. on Principles of Distributed Computing*, Calgary, AB (Aug. 1986), 52–62.

[Blak79] **Blakley, G.** "Safeguarding Cryptographic Keys." *Proc. AFIPS 1979 National Computer Conference*, New York, NY (June 1979), 313–317.

[BlMe85] **Blakley, G.** and **Meadows, C.** "A Database Encryption Scheme Which Allows the Computation of Statistics Using Encrypted Data." *Proc. IEEE Symposium on Computer Security and Privacy*, Oakland, CA (Apr. 1985), 116–122.

[BrCr86] **Brassard, G.** and **Crepeau, C.** "Zero-Knowledge Simulation of Boolean Circuits." *Crypto '86*, Santa Barbara, CA (Aug. 1986).

[CGMA85] **Chor, B., Goldwasser, S., Micali, S.,** and **Awerbuch, B.** "Verifiable Secret Sharing and Achieving Simultaneity in the Presence of Faults." *Proc.* 26th *IEEE Symp. on Foundations of Computer Science*, Portland, OR (Oct. 1985), 383–395.

[Chau81] **Chaum, D.** "Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms." *Comm. ACM 24*, 2, (Feb. 1981), 84–88.

[Chau86a] **Chaum, D.** "Demonstrating that a Public Predicate can be Satisfied Without Revealing Any Information About How." *Crypto '86*, Santa Barbara, CA (Aug. 1986).

[Chau86b] **Chaum, D.** "Elections with Unconditionally Secret-Ballots and Disruption Equivalent to Breaking RSA." *unpublished manuscript* (1986).

[CoFi85] **Cohen, J.** and **Fischer, M.** "A Robust and Verifiable Cryptographically Secure Election Scheme." *Proc.* 26th *IEEE Symp. on Foundations of Computer Science*, Portland, OR (Oct. 1985), 372–382.

[Cohe86] **Cohen, J.** "Improving Privacy in Cryptographic Elections." *TR-454, Yale University, Department of Computer Science*, New Haven, CT (Feb. 1986).

[COS86] **Coppersmith, D., Odlyzko, A.,** and **Schroeppel, R.** "Discrete Logarithms in $GF(p)$." *Algorithmica*, 1 (1986), 1–15.

[DiHe76] **Diffie, W.** and **Hellman, M.** "New Directions in Cryptography." *IEEE Trans. on Information Theory 22*, 6, (Nov. 1976), 644–654.

[DLM82] **DeMillo, R., Lynch, N.** and **Merritt, M.** "Cryptographic Protocols." *Proc.* 14th *ACM Symp. on Theory of Computing*, San Francisco, CA (May 1982), 383–400.

[Feig85] **Feigenbaum, J.** "Encrypting Problem Instances or Can You Take Advantage of Someone Without Having to Trust Him", *Proc. Crypto '85*, Santa Barbara, CA (Aug. 1985), 477–488. Published as *Advances in Cryptology*, ed. by H. Williams in *Lecture Notes in Computer Science*, vol. 218, ed. by G. Goos and J. Hartmanis. Springer-Verlag, New York (1985).

[Fisc83] **Fischer, M.** "The Consensus Problem in Unreliable Distributed Systems", *Proc. 1983 International FCT-Conference*, Borgholm, Sweeden (Aug. 1983), 127–140. Published as *Foundations of Computation Theory*, ed. by M. Karpinski in *Lecture Notes in Computer Science*, vol. 158, ed. by G. Goos and J. Hartmanis. Springer-Verlag, New York (1983).

[FMR84] **Fischer, M., Micali, S.**, and **Rackoff, C.** "A Secure Protocol for the Oblivious Transfer." Presented at *Eurocrypt '84*, Paris, France (Apr. 1984). (Not in proceedings.)

[Fort87] **Fortnow, L.** "The Complexity of Perfect Zero-Knowledge." *Proc.* 19th *ACM Symp. on Theory of Computing*, New York, NY (May 1987), 204–209.

[Gaus01] **Gauss, C.** *Disquisitiones Arithmeticae.* (1801). Translated by Arthur A. Clarke and published by Yale University Press, New Haven (1966). Reprinted by Springer-Verlag, New York (1986).

[GHY85] **Galil, Z., Haber, S.**, and **Yung, M.** "A Private Interactive Test of a Boolean Predicate and Minimum-Knowledge Public-Key Cryptosystems." *Proc.* 26th *IEEE Symp. on Foundations of Computer Science*, Portland, OR (Oct. 1985), 372–382.

[GMR85] **Goldwasser, S., Micali, S.**, and **Rackoff C.** "The Knowledge Complexity of Interactive Proof-Systems." *Proc.* 17th *ACM Symp. on Theory of Computing*, Providence, RI (May 1985), 291–304.

[GMT82] **Goldwasser, S., Micali, S.,** and **Tong, P.** "Why and How to Establish a Private Code On a Public Network." *Proc.* 23rd *IEEE Symp. on Foundations of Computer Science*, Chicago, IL (Nov. 1982), 134–144.

[GMW86] **Goldreich, O., Micali, S.,** and **Wigderson, A.** "Proofs that Yield Nothing But their Validity and a Methodology of Cryptographic Protocol Design." *Proc.* 27th *IEEE Symp. on Foundations of Computer Science*, Toronto, ON (Oct. 1986), 174–186.

[GMW87] **Goldreich, O., Micali, S.,** and **Wigderson, A.** "How to Play Any Mental Game or A Completeness Theorem for Protocols with Honest Majority." *Proc.* 19th *ACM Symp. on Theory of Computing*, New York, NY (May 1987), 218–203.

[GoKi86] **Goldwasser, S.** and **Kilian, J.** "Almost All Primes Can be Quickly Certified." *Proc.* 18th *ACM Symp. on Theory of Computing*, Berkeley, CA (May 1986), 316–329.

[GoMi82] **Goldwasser, S.** and **Micali, S.** "Probabilistic Encryption & How to Play Mental Poker, Keeping Secret All Partial Information." *Proc.* 14th *ACM Symp. on Theory of Computing*, San Francisco, CA (May 1982), 365–377.

[GoMi83] **Goldwasser, S.** and **Micali, S.** "Proofs With Untrusted Oracles." *Unpublished Manuscript* (May 1983).

[GoMi84] **Goldwasser, S.** and **Micali, S.** "Probabilistic Encryption." *J. Comp. Sys. Sci. 28*, (1984), 270–299.

[GoSi86] **Goldwasser, S.** and **Sipser, M.** "Private Coins versus Public Coins in Interactive Proof Systems." *Proc.* 18th *ACM Symp. on Theory of Computing*, Berkeley, CA (May 1986), 59–68.

[Koth84] **Kothari, S.** "Generalized Linear Threshold Scheme." *Proc. Crypto '84*, Santa Barbara, CA (Aug. 1984), 231–241. Published as *Advances in Cryptology*, ed. by G. Blakely and D. Chaum in *Lecture Notes in Computer Science*, vol. 196, ed. by G. Goos and J. Hartmanis. Springer-Verlag, New York (1985).

[Kran86] **Kranakis, E.** *Primality and Cryptography.* John Wiley and Sons, New York (1986).

[MaAd78] **Manders, K.** and **Adleman, L.** "NP-Complete Decision Problems for Binary Quadratics." *J. Comp. Sys. Sci. 16*, (1978), 168–184.

[Merr83] **Merritt, M.** "Cryptographic Protocols." Ph.D. Thesis presented at *Georgia Institute of Technology* (Feb. 1983).

[Mill75] **Miller, G.** "Riemann's Hypothesis and Tests for Primality." Research Report Cs-75-27. Department of Computer Science. University of Waterloo. Waterloo, ON (Oct. 1975). (Abridged version in *Proc.* 7th *ACM Symp. on Theory of Computing*, Albuquerque, NM (May 1975), 234–239.)

[Mill76] **Miller, G.** "Riemann's Hypothesis and Tests for Primality." *J. Comp. Sys. Sci. 13*, (1976), 300–317.

[NiZu72] **Niven, I.** and **Zuckerman, H.** *An Introduction to the Theory of Numbers*, 3rd ed. John Wiley and Sons, New York (1972).

[PoHe78] **Pohlig, S.** and **Hellman, M.** "An Improved Algorithm for Computing Logarithms Over GF(2) and Its Cryptographic Significance." *IEEE Trans. on Information Theory 24*, 1 (Jan. 1978), 106–110.

[Rabi79] **Rabin, M.** "Digitalized Signatures and Public-key Functions as Intractable as Factorization." MIT/LCS/TR-212. MIT Technical Report (Jan. 1979).

[Rabi80] **Rabin, M.** "Probabilistic Algorithms in Finite Fields." *SIAM Journal on Computing 9*, 2 (May 1980), 273–280.

[Rabi83a] **Rabin, M.** "Transaction Protection by Beacons." *J. Comp. Sys. Sci. 27*, 2 (Oct. 1983), 256–267.

[Rabi83b] **Rabin, M.** "Randomized Byzantine Generals." *Proc.* 24th *IEEE Symp. on Foundations of Computer Science*, Tucson, AZ (Nov. 1983), 403–409.

[RAD78] **Rivest, R., Adleman, L.,** and **Dertouzos, M.** "On Data Banks and Privacy Homomorphisms." *Foundations of Secure Computation*, ed. by R. A. DeMillo, et. al. Academic Press, New York, (1978), 169–179.

[RSA78] **Rivest, R., Shamir, A.,** and **Adleman, L.** "A Method for Obtaining Digitial Signatures and Public-key Cryptosystems." *Comm. ACM 21*, 2 (Feb. 1978), 120–126.

[Sham79] **Shamir, A.** "How to Share a Secret." *Comm. ACM 22*, 11 (Nov. 1979), 612–613.

[SoSt77] **Solovay, R.** and **Strassen, V.** "A Fast Monte-Carlo Test for Primality." *SIAM Journal on Computing 6*, (1977), 84–85.

[Stra80] **Straffin, P.** *Topics in the Theory of Voting.* Birkhäuser, Boston, (1980).

[Yao82a] **Yao, A.** "Protocols for Secure Computations." *Proc.* 23rd *IEEE Symp. on Foundations of Computer Science*, Chicago, IL (Nov. 1982), 160–164.

[Yao82b] **Yao, A.** "Theory and Applications of Trapdoor Functions." *Proc.* 23rd *IEEE Symp. on Foundations of Computer Science*, Chicago, IL (Nov. 1982), 80–91.