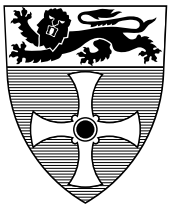
UNIVERSITY OF
NEWCASTLE



University of Newcastle upon Tyne

COMPUTING SCIENCE

The Computer Ate my Vote

P. Y. A. Ryan

TECHNICAL REPORT SERIES

No. CS-TR-988 November, 2006

TECHNICAL REPORT SERIES

No. CS-TR-988 **November, 2006**

The Computer Ate my Vote

P. Y. A. Ryan.

Abstract

I describe the challenges involved in designing and evaluating high assurance, verifiable voting systems. I describe the Pret a Voter scheme that provides voter-verifiablity with minimal trust in officials, software etc. I also describe a number of threats against voting schemes and the extent to which they apply to the basic Pret a Voter scheme. Some enhancements to the scheme designed to counter those threats to which Pret a Voter is potentially vulnerable are presented.

Printed and published by the University of Newcastle upon Tyne,
Computing Science, Claremont Tower, Claremont Road,
Newcastle upon Tyne, NE1 7RU, England.

Bibliographical details

RYAN, P. Y. A.

The Computer Ate my Vote
[By] P. Y. A. Ryan.

Newcastle upon Tyne: University of Newcastle upon Tyne: Computing Science, 2006.

(University of Newcastle upon Tyne, Computing Science, Technical Report Series, No. CS-TR-988)

Added entries

UNIVERSITY OF NEWCASTLE UPON TYNE
Computing Science. Technical Report Series. CS-TR-988

Abstract

I describe the challenges involved in designing and evaluating high assurance, verifiable voting systems. I describe the Pret a Voter scheme that provides voter-verifiablity with minimal trust in officials, software etc. I also describe a number of threats against voting schemes and the extent to which they apply to the basic Pret a Voter scheme. Some enhancements to the scheme designed to counter those threats to which Pret a Voter is potentially vulnerable are presented.

About the author

Peter Ryan is a Professor of CSR. He is responsible for the security and privacy aspects of the DIRC program and is involved in the European MAFTIA project. Prior to joining the CSR, he conducted research in formal methods and information assurance at GCHQ, CESG, DERA, SRI Cambridge, the Norwegian Computing Centre Oslo and the Software Engineering Institute, Carnegie Mellon University. Before migrating into information assurance he was a theoretical physicist and holds a BSc in Theoretical Physics and a PhD in Mathematical Physics from the University of London for research in quantum gravity. He has published numerous articles; the most recent being "Mathematical Models of Computer Security," a chapter in LNCS 2171, is based on lectures given at the FOSAD 2000 Summer School. He is co-author of the book "Modelling and Analysis of Security Protocols," Pearson 2001.

Suggested keywords

VERIFIABLE VOTING,
CRYPTOGRAPHY,
BALLOT PRIVACY

1

The Computer Ate my Vote*

P Y A Ryan

Newcastle University

1.1 Introduction

Voting in the UK is largely conducted using the time-honoured process of marking a paper ballot with a stubby pencil in the privacy of a little plywood booth. Counting is by hand in the presence of various observers and it is generally assumed that the process is reasonably trustworthy and that any corruption or fraud is too insignificant to affect the outcome. As a result, we have a tendency to take our democracy and elections for granted.

This complacency has been shaken in recent years by, amongst other events, the problems with the 2000 and 2004 presidential elections in the US and, closer to home, trials in the UK of postal voting (absentee voting in US parlance). The latter revealed evidence of corruption sufficiently serious to result in prosecution of a number of councillors. The election procedures employed were described by one of the investigating judges as "sufficiently bad to disgrace a minor banana republic". A recent poll by market researchers, Ipsos, indicated that 19% of the UK electorate distrusts the UK electoral system.

We see that, in reality, this democracy that we tend to take for granted is a fragile and precious attribute of civilisation. From the dawn of democracy the Ancient Greeks recognised that people would attempt to corrupt the outcome of elections and used primitive but ingenious technological devices to shift the trust away from people, i.e. officials, to mechanical devices - see for example "Ancient Greek Gadgets and Machine" by Grumbaugh [18].

In the US various technologies for voting have been deployed for over a century, starting with lever machines in 1897, followed by punch cards, optical scanners, touch screens etc. Interestingly, these innovations appear to have been largely prompted by high occurrence of fraud with paper ballots. The

* Chapter to appear in "Formal Methods: State of the Art and New Directions", edited by P.P. Boca, J.P. Bowen J.I. Siddiqi to be published by Springer in 2007

excellent book by Gumbel, "Steal this Vote" [19], documents an extraordinary litany of instances of corruption and ingenious techniques to corrupt every voting system that has ever been deployed in the US. Recent reports, like the Johns Hopkins report, [26] and more recently the Princeton report [15], on the Diebold touch screen devices, demonstrate how vulnerable a poorly designed electronic voting system is to virtually undetectable corruption.

Against this backdrop, one might reasonably ask: why not just stick with the tried and tested pencil and paper? This reaction has much to commend it: there do appear to be good grounds to believe that to pull off large scale, undetected fraud would be difficult. The fact that anyone can apply to be an observer during the parts of the process gives some degree of transparency. The current UK system is certainly not perfect but it does appear to quite robust and commands a reasonable degree of confidence. It does have weaknesses and requires significant levels of trust in the officials and processes.

Sticking with the traditional techniques for conducting elections appears not to be an option. There is pressure, mainly from politicians, for the adoption of electronic means for various stages in the processing of votes, apparently in the belief that the convenience of remote, e.g.: internet, phone voting, etc. will encourage higher participation. It is argued that the use of touch screens may help guide voters through the more complex, multi-race ballots presented to voters, as with many US elections. Appropriate use of technology may enable voters with disabilities to vote in privacy. Electronic counting potentially could help with speed, accuracy and efficiency. There seems also to be a sentiment that continuing to use 18th century methods in the 21st century feels a little archaic and unworthy of a modern democracy.

There are thus numerous reasons to explore more technologically advanced voting systems. For a cryptographer, the most compelling reason to explore alternative approaches, aside from the intellectual challenges, is the potential to create schemes that will provide far greater levels of assurance of accuracy and ballot secrecy whilst removing the need to trust suppliers, software or officials. Schemes like those described below may provide a sound basis on which to restore faith in the mechanisms of democracy. These schemes can be thought of as striving to place democracy on the firm foundations of mathematics. I think that the Ancient Greeks would have looked favourably on such an endeavour!

The material presented here is based on previously published material at [35, 8, 38, 36].

1.2 The Challenge

The challenge is to design systems that provide high assurance of the accuracy of the outcome, whilst at the same time guaranteeing the secrecy of ballots.

The problem of course is that these two requirements are in conflict. Each taken in isolation would be trivial to achieve. In the absence of the secrecy requirement, a simple show of hands guarantees accuracy. Equally, if accuracy is not required, then secrecy is trivial: a constant function achieves this, i.e., the result is totally independent of the inputs.

Similarly, if we are prepared to place complete trust in the processes that collect and count the votes then again there is no real problem. This, in essence, is how the current UK systems works and, whilst there seem to be good grounds to believe that such trust is well-placed for the time being, there are contexts in which such trust would be wholly inappropriate. Who is to say that the UK will remain as benign a political environment? It would seem prudent therefore to explore voting systems that do not necessitate such trust.

The goal then is to provide assurance of accuracy and secrecy without having to place any trust in the devices, processes and officials that conduct the election. Put differently, we strive to design systems that will be resilient to insider threats as well as to outsider threats.

A fundamental feature of secret ballot voting systems that distinguishes them from conventional dependable systems is that there is no extrinsic way to characterise the correctness of the outcome. By definition, there is no god-like view that can determine if the outcome is correct and it is thus possible for an election system to fail in a way that is not manifest. This is in contrast with, say, an algorithm for computing square roots, or for that matter, avionics software. The correctness of the outcome is clearly defined and independently verifiable, or, turning this around, failures of such systems will be manifest. Failures of voting software could go unnoticed and, even if suspected, may be impossible to demonstrate.

As if all this were not challenging enough, we must also take account of all the socio-technical aspects of the problem: our systems must be transparent and simple enough to gain a reasonable degree of usability, public understanding and trust. A technically perfect solution that fails to gain the confidence of the electorate is not a viable solution. Furthermore, we must take account of the surrounding socio-technical systems that maintain the electoral register, authenticate voters, officials and scrutineers who supervise and observe the process, etc.

To a cryptographer, with a rather paranoid view of the world, a voting system is a highly hostile environment: the system is trying to cheat the voters, the voters are trying to circumvent the system, officials are trying to manipulate the system, coercers are attempting to influence voters, and voters trying to cheat coercers. Curiously, the last of these is one that we want to enable! Of course, all this is very much a worst case analysis, but if we can make our systems resistant against such adversaries then they should also be robust in less hostile environments. Like Hollywood producers, cryptographers like their adversaries to be imbued with unlimited malevolence and cunning.

1.3 Assumptions

Voting systems are large socio-technical systems comprising not only various technical components but voting officials, voters, campaigners, potential coercers etc. The process of an election goes far beyond merely collecting and counting votes and has social, political, legal and psychological aspects. For the purposes of this chapter, we will make a number of rather sweeping assumptions and we will draw rather precise boundaries around the portion of the voting system that we will discuss. In practice of course, all of these assumptions have to be closely examined and the scheme evaluated in the context of the wider socio-technical system.

We will confine our attention to the more technical processes and mechanisms that capture and count the votes. We will assume that an accurate register of eligible voters is maintained and that suitable access to this database is ensured throughout the period of voting. We will assume that suitable mechanisms are in place to authenticate voters and ensure that they do not vote more than once.

1.4 Voting System Requirements

At the most abstract level, we require that elections be free and fair. Freeness can be taken to mean that, throughout the period of the election, legitimate members of the electorate are able to express their intent without hindrance or undue influence. Quite what constitutes "undue influence" is a rather delicate matter. Some would contend that political manipulation of the media, bandying about of vacuous promises and so on constitute undue influence. We will leave such matters to the political scientists and concentrate on eliminating the rather crude but better defined threats of to vote buying and coercion.

Fairness is taken to mean that every voter is able to cast at most one vote and that all votes cast are accurately counted and reflected in the final tally. These rather vague statements are mapped down to more precise requirements on the various components of the socio-technical system.

There is no universal consensus as to voting system requirements and in any case, they will vary from application to application and according to jurisdiction. Here we informally describe the commonly accepted requirements of the vote capture and counting components of the system.

The primary goal of a voting system is integrity (a.k.a. accuracy): all legitimately cast votes should be accurately included in the count. Furthermore, accuracy, like justice, should not only be done but also be seen to be done. Thus we will strive for *verifiability*, i.e: mechanisms that demonstrate that the count is accurate. Often, cryptographic schemes provide unconditional

integrity, that is, they provide guarantees of integrity that do not depend on computational assumptions. In other words, integrity is guaranteed even against adversaries with unbounded computational power.

We will require that a voter's choice is kept secret, often termed *ballot secrecy* or *ballot privacy*. This requirement stems from the need to avoid threats of vote buying or coercion. Some forms of voting call for accountability, for example parliamentary voting, and hence secrecy is not required.

In fact, absolute ballot secrecy is not required or provided by the UK voting system. Rather, UK electoral law requires that it be possible for the authorities, with appropriate court orders, etc, to trace ballots back to the (claimed) identity of the person who cast the ballot. It is still expected that no one other than appropriate authorities be able to establish how a particular voter voted[2]. We will take ballot secrecy as a fundamental requirement in what follows.

A rather novel requirement, not feasible in conventional systems, is that of *voter-verifiability*. Voters are able to confirm that their vote is accurately included in the count and, if not, to prove this to a judge. At the same time, the voter is not able to prove to a third party which way they actually voted. At first glance this seems impossible, but the schemes that we describe below do realise this requirement. Modern cryptography regularly makes the seemingly impossible quite routine!

Besides the above technical requirements, voting systems must also be cost-efficient, easy for voters to use and sufficiently simple to gain a sufficient degree of public understanding and trust. Discussion of these requirements is outside the scope of this chapter, but we remark that the systems described here are aimed at being as conceptually simple as possible.

1.5 Sources of Assurance

It is important to understand the starkly contrasting nature of the assurance provided by different technologies. The most naive approach is simply to trust the technology or process. In some contexts this is probably quite reasonable e.g: using an ATM, where fraud is detectable and forms of insurance and redress are available. Voting is definitely not in this category: elections are essentially one-off events, fraud may go undetected and the consequences are potentially global.

[2] Cryptographic schemes can support a requirement to reveal links between receipts and votes, but this requires the cooperation of a predefined threshold number of servers. These servers could be controlled by independent authorities and organisations. Thus, the checks and balances preventing abuses of such a capability are far more transparent and accountable than at present.

Assurance of correct behaviour of any system may be derived in various ways. It might be based on (claims of) prior evaluation, verification and testing of the system. Such analysis seeks to demonstrate that all possible system executions will satisfy the requirements. On the other hand, assurance might be derived from run-time monitoring of the system as it executes and end-to-end checking of outputs, ensuring that any departure from correct behaviour is detected.

Both approaches have a useful role to play in certain contexts and typically both must be employed, but the emphasis may vary. For systems in which it may be hard to recover from a malfunction and you really want confidence about future performance, e.g. an avionics system, emphasis must be placed on the former. The drawback is that complete and correct analysis is extremely difficult to achieve and, even if it were achieved, system degradation or upgrades, alteration of the code etc may all serve to invalidate any analysis. Such assurance is thus very fragile.

End-to-end checking is independent of the software and hardware implementation and so is more robust: as long as we have a precise definition of correct behaviour and can implement a suitable monitoring mechanism, we can be confident that any errors will be detected and rectified. We are thus no longer prey to the problems of incomplete analysis or testing, or the mutability of software etc. In the case of voting systems, we argue that assurance of integrity must be end-to-end. The correctness of the outcome must be mathematically verifiable, rather as the correctness of a square root algorithm is independently verifiable. To borrow Benaloh's phrase: we should verify the election, not the system.

This is not to say that evaluation and testing of the system should be ignored, but just that the accuracy of the outcome must not be contingent on the correctness, coverage and continuing validity of such evaluation. Naturally it is essential that an election run smoothly and not be plagued by malfunctions and technology glitches, so careful testing and evaluation will be also be essential.

1.5.1 Assurance of Privacy

The discussion above applies to the sources of assurance of integrity. Secrecy and privacy properties are of quite a different nature to integrity properties. Integrity is a safety property, i.e: can be characterised by defining what are acceptable system behaviours. Secrecy properties, by contrast, are usually defined in terms of the entire set of possible behaviours rather than individual behaviours. Thus, for example, the secrecy provided by a stream cipher depends on it being effectively indistinguishable to an adversary from a device able to generate all possible random streams with equal probability (i.e: a

one-time-pad). This is clearly not a property of any given behaviour but of the ensemble of possible system behaviours.

Consequently, it is typically not possible to detect failures with respect to a secrecy property by monitoring an individual system execution, rather as it is not sensible to ask if a given bit sequence is random. It also tends to be much harder to recover from secrecy failures (...it is hard to get the toothpaste back in the tube as one US judge phrased it). Consequently, for secrecy, we need to place more emphasis on prior verification than we do for integrity.

1.6 Verifiable Voting Schemes

Many cryptographic schemes that seek to provide high levels of assurance of accuracy and secrecy have been proposed. Typically these strive to minimise the need to place trust in the implementations and seek assurance through maximal transparency. In accordance with the principle of no security through obscurity, the details of these schemes are laid bare to universal scrutiny, so that any flaws in the design can be detected before deployment. Furthermore, the integrity of the election rests now on the validity of the mathematical arguments rather than on mutable implementations.

This is a key and very subtle point: if the implementation malfunctions or is subverted, this should be detected at run-time by the auditing procedures. All the computations performed during the auditing phase are against publicly known functions and hence independently verifiable by anyone who cares to take the trouble. Furthermore, such verification is performed on data that is frozen and committed rather than on observation of an ephemeral process.

1.7 Related Work

There is a large and rapidly growing literature on cryptographic voting schemes and it is not appropriate to attempt to survey it all here. We will just mention the most closely related literature, concentrating on schemes designed for the supervised rather than remote context. A more complete survey can be found in [3], and for anonymity mechanisms, mixes etc see the anonymity bibliography, [1].

The first suggestion that cryptographic techniques could be applied to voting systems appears to be Chaum's 1981 paper on anonymising mixes, [6]. Benaloh and Tunistra, [5], introduce the notion of coercion-resistance along with a scheme using homomorphic tabulation that satisfies it. Later, Chaum [7] and Neff [28] introduced schemes that could be regarded as more practical

than previous schemes. The original Prêt à Voter scheme, [34] and [33] was inspired by the Chaum scheme, replacing the visual cryptography by the candidate permutation concept. Chaum has subsequently adopted this concept in his new PunchScan scheme, [16]. Recently, Rivest has proposed *ThreeBallot*, [32] that provides voter-verifiability without using cryptography.

1.8 Cryptographic primitives

Firstly we introduce some cryptographic primitives that will be used later. In the interests of space, and not bogging the reader down in unnecessary technical detail, we give very superficial descriptions, enough; we hope, to make their role clear. For more detailed descriptions consult any book on modern cryptography, e.g: Stinson [39]. Note also that this overview is not intended to be exhaustive: we stick to those required for Prêt à Voter. We assume familiarity with the basics of number theory. No offence will be taken if you are familiar with modern cryptography and choose to skip this section.

1.8.1 Public key cryptography

Some thirty years ago, the rather shadowy discipline of cryptography witnessed a major revolution: in 1976 Diffie and Hellman published a paper, [13] that, for the first time in the open literature, proposed the possibility of public key cryptography. Arguably, this paper did for cryptography what Einstein's 1905 special relativity paper did for theoretical physics. Prior to 1976, it was implicitly assumed that secret communication could only be achieved between parties who already shared secret material. Diffie and Hellman overturned this assumption by suggesting that it should be possible for complete strangers who did not share any prior secrets to communicate secretly over open channels.

At first glance this suggestion seems absurd. To realise it requires the concept of one-way-functions: functions that are easy to compute in one direction but intractable to compute in the other. One such function is that of multiplication: multiplying a pair of given numbers is straightforward yet, given a number, finding its prime factors is in general extremely difficult.

Using the concept of one way functions, it is possible to implement encryption algorithms with the following features: Anne creates a pair of keys, one that is made public, the other Anne keeps secret. The public key can be used by anyone to encrypt a message for Anne, but decryption can only be performed with knowledge of the secret key. Thus, Anne, and only Anne, can decrypt such messages. From a classical cryptography perspective this seems impossible: encryption and decryption keys are closely related, often identical.

The key point is that knowledge of the public encryption key does not entail knowledge of the secret key.

The ideas of public key were in fact anticipated, in secret, at the Government Communications Headquarters (GCHQ). In 1970 James Ellis came up with the idea of *non-secret encryption* and in 1973, Clifford Cocks came up with an implementation of the concept that is essentially the same as the RSA algorithm we describe below. None of this was available in the open literature until recently, so it is fair to say that these concepts were independently re-invented.

1.8.2 RSA Encryption

The first realisation of the public key concept to appear in the open literature is that proposed by Rivest, Shamir and Adleman, [30]. The security of the scheme rests on the assumed difficulty of factorising. First, Anne first identifies two large prime numbers, p and q. By "large" here we typically mean of the order of a hundred digits. Given that factorisation is hard, it would appear to be hard to find large primes. However, efficient algorithms exist for finding primes that do not involve factorising. Typically such algorithms do not yield primes with certainty but with arbitrarily high probabilities [3].

Let $n := p \cdot q$. Now Anne chooses an encryption exponent e such that $gcd(e, n) = 1$, i.e., e must be co-prime to n. Now she finds d such that:

$$e \cdot d = 1 \pmod{\phi(n)}$$

Where $\phi(n)$ is Euler's totient function, in this case $\phi(n) = (p-1) \cdot (q-1)$. This computation can be performed very efficiently using the extended Euclidean algorithm. Now Anne can make n and e public whilst she keeps d (and p, q and $\phi(n)$) secret. Now Bob can encrypt a message m say, which is suitably encoded as a number in Z_n, the set of residue classes modulo n, as:

$$c := m^e \pmod{n}$$

Anne can decrypt this by computing:

$$c^d = m^{e \cdot d} = m \pmod{n}$$

[3] A simple example of such an algorithm is based on Fermat's Little Theorem: if p is prime, then $\forall a \in Z_p^*, a^{p-1} = 1 \pmod{p}$. Thus, for a putative prime p, we choose a set of a's at random and check whether Fermat's congruence holds for all of them. If it does then we have, with high probability identified a prime p.

To see why this works, note that, by construction, $e \cdot d = r \cdot \phi(n) + 1$ for some $r \in Z$. So, by Euler's generalisation of Fermat's Little Theorem:

$$\forall a \in Z_n^* : a^{\phi(n)} = 1 \pmod{n}$$

Where Z_n^* is the set of residue classes modulo n with multiplicative inverse, i.e., $Z_n^* = \{a \in Z_n | gcd(a, n) = 1\}$.

It follows that:

$$m^{e \cdot d} = m^{\phi(n) \cdot r + 1} = m^{\phi(n) \cdot r} \cdot m^1 = 1 \cdot m = m \pmod{n}$$

as required.

Computing exponents in a finite field can be done very efficiently using repeated squaring and multiplication. Computing d from e in is straightforward with the knowledge of the factorisation of n, and hence of $\phi(n)$. Without knowing the factorisation there is no known, efficient way to find d short of essentially exhaustive search. No efficient algorithm is known to factorise composite numbers that are the products of such large primes. Hence it is believed that it is intractable to compute d given only n and e, i.e, without knowledge of the factorisation of n. So, as long as Anne guards the secret value d, and the factorisation of n, she alone can decrypt messages encrypted using her public key. Anyone knowing n and e can encrypt messages for Anne. Anne of course started with the primes p and q and so is not faced with the problem of discovering them by factorising n.

1.8.3 ElGamal Encryption

RSA as described above is deterministic; encrypting a given plaintext twice yields the same ciphertext. We now describe a couple of randomising algorithms for which repeatedly encrypting a given plaintext will yield different ciphertexts each time.

The first algorithm is due to ElGamal, [14]. Anne finds a large prime p and a primitive element α of the multiplicative group Z_p^* (α is said to be primitive if every element of Z_p^* can be expressed as a power of α). Anne chooses a random k from Z_p and computes:

$$\beta := \alpha^k \pmod{p}$$

The public keys are p, α and β, k is kept secret. Encryption of m yields a pair of terms computed thus:

$$c := (y_1, y_2) := (\alpha^r, m \cdot \beta^r) \pmod{p}$$

where r is chosen at random from Z_p. Anne, with the knowledge of k, is able to decrypt as follows:

$$m = y_2 / y_1^k \pmod{p}$$

The security of ElGamal rests on the presumed difficulty of taking discrete logs in a finite field. We remarked earlier that taking exponents is perfectly tractable, but the inverse operation of taking discrete logs is believed to be intractable. Thus, recovering the secret k exponent from knowledge of p, α and β is thought to be intractable.

A randomising algorithm like ElGamal allows the possibility of re-encryption: anyone who knows the public keys can re-randomise the original encryption with a new random value r':

$$(y_1', y_2') := (\alpha^{r'} \cdot y_1, \beta^{r'} \cdot y_2)$$

which gives:

$$(y_1', y_2') := (\alpha^{r'+r}, \beta^{r'+r} \cdot m)$$

Clearly, this is equivalent to simply encrypting m with the randomisation $r + r'$ and decryption is performed exactly as before. We will see the utility of re-encryption when we come to describe anonymising mixes. Note that, crucially, the device performing the re-encryption does not use any secret keys and at no point in the re-encryption process is the plaintext revealed.

1.8.4 Paillier encryption

Paillier is another randomising algorithm that, due to its homomorphic properties, is very handy in voting applications, [29]. Key generation proceeds as follows: firstly generate an RSA integer $n = p$ with p and q large primes and compute the Carmichael function of n: $\lambda := lcm(p-1, q-1)$. Find a generator g of $Z_{n^2}^*$ such that $g = 1 \pmod{n}$. The public key, (n, g) is published whilst λ forms the secret key.

The encryption of a message $m \in Z_n$ is computed as:

$$c = \mathcal{E}(m, r) = g^m r^n \pmod{n^2}$$

Where r is a freshly generated random value drawn from Z_n^*.

Decryption is given by:

$$m = \frac{L(c^{\lambda} \pmod{n^2})}{L(g^{\lambda} \pmod{n^2})} \pmod{n}$$

Where, for convenience, we have defined $L(x) := (x-1)/n$.

We won't go into exactly why this rather surprising formula works. The key ingredient is that fact that:

$$\forall a \in Z^*_{n^2} : a^{n.\lambda(n)} = 1 \pmod{n^2}$$

Judicious use of the binomial theorem to expand g, expressed as $1 + k \cdot n$, and noting that terms with n^2 or higher order in n vanish taken modulo n^2, serves to move the plaintext m term from the exponent to a linear term.

Due to the way that the plaintext is carried in the exponent, the Paillier algorithm enjoys the homomorphic property:

$$\mathcal{E}_P(a; r) \times \mathcal{E}_P(b; s) = \mathcal{E}_P(a + b; r \times s)$$

Where $\mathcal{E}_P(x; r)$ denotes the encryption of x and randomisation r. That is, the product of the encryption of two plaintexts equals the encryption of the sum of the plaintexts.

1.8.5 Threshold Cryptography

It is often important not to have to depend on a single entity to perform cryptographic operations, such as encryption, decryption, signing, etc. This prompts the development of techniques and algorithms to distribute the knowledge of the secret key amongst a set of entities Φ in such a way that only a quorum of Φ can perform the operation. An (m, k) threshold scheme for example allows k or more entities from a set of size $m, (k < m)$, to collaborate to perform the operation. Any smaller set of size $< k$ will be unable to perform the operation, nor will they be able to obtain any information about the secret key.

1.8.6 Anonymising Mixes

Anonymising mixes were first proposed by Chaum [6] and play a key role in many voting schemes. They come in two flavours: decryption and re-encryption, but hybrids are also possible.

In decryption mixes, a deterministic algorithm such as RSA is used. The plaintexts are encrypted in layers under the public keys of a set of mix servers. The batch of encrypted terms is passed to the first server that strips off the outer layer of encryption and performs a secret shuffle of the resulting terms. The resulting, partially decrypted, shuffled terms are passed on to the next server that strips off a further layer, shuffles the resulting terms and passes this along to the next. Thus, the batch of encrypted terms is put through a series of such mixes that progressively decrypt and shuffle. At the end of these mixes, the raw, decrypted votes pop out but with any link to the original terms obliterated by the multiple shuffles.

For re-encryption mixes, a randomising algorithm such as ElGamal or Paillier is used. Instead of striping off layers of encryption at each step of the mix, the mix servers re-randomise the encryption. The mix servers do not hold any secret keys: re-randomisation can be performed by any entity that knows the public key. A sequence of such mixes can then be followed by a (threshold) decryption by servers that hold the appropriate secret keys.

The fact that all the terms passing through a mix undergo a transformation, either decryption of re-encryption, at each stage ensures that they cannot be simply traced through the mix by pattern matching, as long as at least one of the servers remains honest.

1.8.7 Homomorphic Tabulation

Anonymising mixes are well-suited to taking a batch of encrypted ballots and outputting the decrypted ballots in a shuffled order. An alternative way to tabulate a batch of encrypted ballots is to use *homomorphic tabulation*. Here, a cryptographic primitive is employed that exhibits algebraic homomorphic properties that enables the count to be performed without needing to decrypt individual ballots. For example, the Paillier algorithm has the property that the product of the encryption of a set of values equals the encryption of the sum of the values:

$$\prod_{i}^{n} \mathcal{E}_P(x_i) = \mathcal{E}_P(\sum_{i}^{n} x_i)$$

This property can be exploited to extract the overall count without having to decrypt individual votes. For a simple yes/no referendum, yes votes are represented by (randomised) encryptions of $+1$, whilst no votes are encryptions of 0. Suppose that n votes are cast. The product of the encryptions of votes is formed and the result decrypted. If the overall sum is greater than $n/2$, the ayes carry it, if the sum is less than $n/2$ the nays have it. For elections with choices of more than 2 candidates or options, more subtle encodings are required, [4].

Tabulation using mixes can thus be thought of as analogous to conventional counting of paper ballots whilst homomorphic tabulation can be thought of as roughly analogous to the operation of lever machines: only the cumulative totals for each candidate are output. Care has to be taken to ensure that ballots are only ever encryptions of the either 0 or 1, otherwise there are dangers of a single ballot seriously skewing the count.

1.8.8 Cut and Choose

Cut-and-choose is a common device to avoid having to trust a device that performs a cryptographic operation, typically encryption. The obvious approach is to require the keys or randomisations to be revealed so that the encryption can be checked. In many situations, notably voting, this is not satisfactory as the proof could be transferred to another party, i.e: a coercer. Verifiying the encrypted term renders it useless in the rest of the protocol [4]. What we want is a way for the verifier to be confident that the encryption is correct without being able to prove this to anyone else.

The name comes from the familiar protocol for ensuring fairness in the sharing out of a cake: one party makes the cut whilst the other gets to choose. The effect is to motivate the first party to try to cut as fairly as possible. Analogously, an encryption device is required to commit to several independent encryptions of the given plaintext. It is then challenged to reveal the keys or randomising factors for all but one, randomly chosen by the requestor. If all of the challenged encryptions prove to be valid then it is a good bet that the un-revealed encryption is valid too and can be used in the rest of the protocol.

1.8.9 Zero-knowledge proofs

Cut-and-chooose protocols involve generating an excess number of ciphertexts, auditing a randomly selected subset and rejecting the audited elements, as their cover has been blown. A more subtle way of establishing confidence that, for example a given ciphertext really is an encryption of a claimed plaintext, is to use *zero-knowledge* proofs.

An interactive zero-knowledge proof (ZKP) is a protocol in which one party, the prover P, demonstrates the truth of a claim or knowledge of a fact to another, the verifier V, without V learning anything other than the truth of the statement. Such protocols typically involve a sequence of random challenges issued by the verifier to the prover.

[4] This reminds me of being puzzled as a kid by talk of "testing the bomb".

A typical example of such a protocol is the Chaum-Pedersen protocol, [9] that is designed to prove plaintext equivalence of a pair of ElGamal encryptions without revealing either the plaintext, the secret key or the re-randomising factor. This situation crops up where a server has performed a re-encryption on an ElGamal ciphertext and wants to prove the correctness without revealing the re-randomisation factor. The proof is reducable to showing that a tuple (α, β, w, u) is a so-called *Decision Diffie-Hellman* (DDH) tuple $(\alpha, \alpha^k, \alpha^x, \alpha^{k.x})$, i.e., $\exists\ x$ and k such that $w = \alpha^x$, $u = \alpha^{x \cdot k}$ and $\beta = \alpha^k$. Here, k is thought of as the secret ElGamal key and x the re-encryption factor. Where the prover P is a mix server demonstrating plaintext equivalence, P will know the re-encryption factor x but not the decryption key k.

The three step protocol follows the standard pattern for ZK proofs: P generates some fresh randomness, s, that serves to blind the secret and makes a commitment. V responds with a random challenge, c, to which P should only be able to respond with the value $t = s + c.x$ verifiable by V, if he really knows the secret value x.

1. $s \in Z_q^* : P \rightarrow V : (a, b,) = (\alpha^s, \beta^s)$
2. $c \in Z_q^* : V \rightarrow P : c$
3. $P \rightarrow V : t = s + c.x$

Now V can check:

$$\alpha^t = a.w^c \quad and \quad \beta^t = b.u^c$$

Informally, we see that the secret, random factor s chosen by P serves to conceal the secret value x from V. If P does not know x, or indeed, the claimed equivalence is false and such an x does not exist, it will be virtually impossible him, aside from an absurdly unlikely lucky guess, to respond to v's challenge value c with a value t that will pass V's checks.

A variant of this protocol can be used to demonstrate the correctness of a claimed decryption of a given ElGamal ciphertext. Again, the proof can reduced to the proof of a DDH tuple. In this case, P knows k but not the randomising factor x so we simply interchange their role in the protocol. Suppose that we have the ElGamal ciphertext $(y_1, y_2) = (\alpha^k, m.\beta^k)$ and P claims that this decrypts to m'. To check that $m = m'$ we require P prove that the tuple $(\alpha, \beta, y_1, y_2/m')$ is a DDH tuple, which it will be iff $m = m'$.

A similar protocol to prove correct decryption of a Paillier ciphertext can be found at [11] in the case in which the prover knows the randomisation. For Paillier it turns out that knowledge of the secret key allows the prover to recover the randomisation as well as the plaintext. Thus there is no need for a separate protocol for the case in which the prover is ignorant of the randomisation. This is in contrast to ElGamal, where knowledge of the secret key does not help recovering the randomisation.

1.8.10 Digital Signatures

These are the digital analogue of conventional signatures. Anne can digitally sign a text M by computing a publicly agreed crypto hash of M and encrypting this under her private key. This encrypted hash is appended to the text M:

$$Sig_A(M) := M, \{Hash(M)\}_{SK_A}$$

If Bob knows Anne's public key, he can verify the signature as follows: he applies Anne's public key to the encrypted term and applies the hash function to the plaintext M term. If the outcome of these two computations agree, he may be confident that the text M was indeed signed by Anne and has not been altered. This assumes that Anne's private key has not been compromised.

1.9 Voter-verifiable Schemes

Cryptography opens up novel and surprising possibilities, in particular the notion of *voter-verifiability*. A system that enjoys voter-verifiability enables voters to confirm to their own satisfaction that their vote is accurately recorded and counted whilst at the same time denying them any way of proving how they voted to a third party. The key idea to achieving voter-verifiability is to provide voters with a physical receipt at the time of casting that carries their vote in encrypted form. Voters are later able to confirm, via a *secure Web Bulletin Board* (WBB) or similar [5], that their receipt has been correctly entered into a robust, anonymising tabulation process. We need mechanisms to ensure that:

- The voter's choice is accurately encoded in the receipt.
- The voter's receipt is accurately posted to the WBB.
- The tabulation process accurately decrypts all posted receipts.

If all three assertions are be demonstrated, then each voter may, by confirming that their receipt is accurately input into the tabulation, be confident that their vote is counted as cast. Furthermore, if their receipt fails to appear correctly on the WBB, they have tangible proof of this. The fact that their vote is encrypted in the receipt ensures that they cannot use it to prove how they voted to a coercer or vote-buyer.

Over the past few years, a number of cryptographic schemes have been proposed to implement this concept and to provide voting systems with high

[5] The list of encrypted receipts could be published for example in The Times

assurance of accuracy and ballot secrecy. Such schemes strive to empower the voters by providing them with the means to help contribute to the dependability of the process. Dependability by the people for the people one might say.

The Prêt à Voter suite of schemes constitutes a class of particularly voter-friendly, voter-verifiable schemes and we outline a simple instance of this class in the next section. Later we will explore a number of threats to which this basic version is vulnerable. This will lead us to develop a number of enhanced versions. It is hoped that this style of presentation will result in gentle introduction to the key ingredients of the Prêt à Voter approach whilst making clear the subtleties involved in designing such schemes.

1.10 Outline of Prêt à Voter

Here we outline the main ingredients of the Prêt à Voter scheme, [35, 8, 38]. The key innovation of the Prêt à Voter approach is the way votes are encoded in a randomised frame of reference, i.e: a randomised candidate list. An important observation about this way of encoding the vote is that, in contrast to previous schemes, there is no need for the voter to communicate their vote to an encryption device. What is encrypted is the information that defines the frame of reference for any given ballot form, and this can be computed in advance. We will return to the significance of this observation later when we discuss the threat model. Incidentally, this encoding has another advantage: the randomisation of the candidate list results in fairness as a fixed ordering tends to favour candidates near the top of the list.

1.10.1 The Voting Ceremony

At the polling station, our voter Anne pre-registers and chooses at random a ballot form from a pile of forms individually sealed in envelopes. Example forms are shown in Figures 1.1 and 1.2. Note that the order of the candidates and the cryptographic values vary from form to form.

In the booth, Anne removes the ballot from its envelope and makes her selection in the usual way by placing a cross in the right hand column against the candidate of choice, or, in the case of a Single Transferable Vote (STV) system for example, she marks her ranking against the candidates. Once the selection has been made, she detaches and discards the left hand strip that carries the candidate order. The remaining right hand strip now constitutes the receipt, as shown in Figure 1.3.

Anne now exits the booth with this receipt, registers with an official and casts her receipt in the presence of the official. The ballot receipt is placed

Candidates	Your vote
Obelix	
Idefix	
Asterix	
Panoramix	
	$7rJ94K$
Destroy	Retain

Fig. 1.1. Prêt à Voter ballot form

Candidates	Your vote
Asterix	
Idefix	
Panoramix	
Obelix	
	$N5077t3$
Destroy	Retain

Fig. 1.2. Another Prêt à Voter ballot form

Your Vote
X
$7rJ94K$
Retain

Fig. 1.3. Prêt à Voter ballot receipt encoding a vote for "Idefix"

against an optical reader or similar device that records the cryptographic value at the bottom of the strip, that we will refer to henceforth as the ballot *onion* and denote Θ, and an index value ι indicating the cell into which the X was marked. A digital signature is computed over $\{\iota, \Theta\}$ and this signature is printed on Anne's receipt.

The digitized copies of the receipts are transmitted to a central tabulation server which posts them to a secure WBB. This is an append-only, publicly visible facility. Only the tabulation server, and later the tabulation tellers, can write to this and, once written, anything posted will remain unchanged. Voters are encouraged to visit this WBB and confirm that their receipt appears correctly and, if their receipt does not appear, or appears incorrectly (i.e., with the X in the wrong position), they can appeal. Note that, as the voters hold physical, authenticated receipts, they have demonstrable grounds for complaint if their receipt fails to appear.

At the time of casting of the vote, in addition to the digital copy of the receipt, it is possible for an additional paper copy to be printed and verified by the voter and officials and added to an encrypted paper audit trail. This is similar to the notion of a *Voter Verified Paper Audit Trail* but with encrypted receipts, the audit trail can be printed on a paper roll like a till receipt without jeopardizing privacy [6]. This is easier to implement a record on a single roll and it is much harder to manipulate than a bundle of ballots. A VEPAT can also serve as the basis for additional checks of correspondence between the audit record and what is posted to the WBB to be performed by independent auditors to supplement voter checks.

It is also possible to have representatives of *Helper Organisations*, [3], at hand at the polling stations. They could offer a receipt verification service: checking digital signatures and possibly checking of posting of receipts to the WBB.

1.10.2 Vote Counting

The value printed on the bottom of the RH column of the ballot forms, that we will refer to as the ballot *onion*, is the key to extraction of the vote. Buried cryptographically in this value is the information needed to reconstruct the candidate order and so interpret the vote value encoded on the receipt. This information is encrypted under the secret keys shared using a threshold scheme amongst a number of tellers. Thus, only a threshold set of the tellers acting in concert are able to reconstruct the candidate order and so interpret the vote value encoded on the receipt.

Each form will have a unique, secret seed value ρ drawn from some seed space S. The candidate permutation is computed using function σ that is publicly agreed prior to the election: : $S \rightarrow \Pi_C$, where Π_C is the set of permutations on the candidate set C. Thus each ballot form may be thought of as a tuple:

$$(\pi, \Theta)$$

where π is the candidate order and the encrypted term $\Theta = \mathcal{E}(\rho))$ is the onion. The ballot is well-formed if and only if $\pi = \sigma(\rho)$. The value ρ is kept secret. A receipt has the form:

$$(\iota, \Theta)$$

[6] With a conventional VVPAT. system with un-encrypted ballots, using a till roll record introduces the possibility of correlation between the order on the roll and the order voters entered the booth.

Where ι is an index value indicating where the voter placed her X or a vector giving her rankings.

After a suitable period, assuming that checks on the posted receipts are satisfactory or at least suitably resolved, we can start the counting process. Here we will describe counting using *anonymising mixes* to guarantee ballot secrecy and *partial random checking* to guarantee correctness. Other approaches are possible, for example, if Paillier encryption is used it would be feasible to use homomorphic tabulation.

We assume that the candidate list information has been encrypted in the ballot onions using a randomising algorithm that supports re-encryption, e.g: ElGamal or Paillier. We also assume for the moment that the index values, indicating the position of the X for example, has been absorbed into the onion term to give a pure ElGamal or Paillier term. Details will be presented a little later. Tabulation proceeds in two phases: first a mixing phase to provide privacy (rather like shaking the ballot box) followed by a decryption phase (unsealing and unlocking the ballot box). The first phase will be performed by a set of *mix tellers*. The mix tellers do not need to know any secret keys in order to perform a re-encryption, only the public key under which the onion encryption was performed.

Suppose that we have j mix tellers, each of which will perform two re-encryption mixes before handing on to the next teller. Requiring each mix teller to perform two mixes is a convenient device to facilitate the audit process, as we will see shortly. More precisely, the first mix teller takes in the batch of receipts, re-encrypts each and then performs a secret shuffle on the resulting batch of terms. The resulting batch of transformed, shuffled terms is posted to the next column of the WBB. The first teller then takes the output of its first mix and repeats the process, but with an independent shuffle, and again posts the result to the next column of the WBB. Once the first teller has finished performing its two mixes, the second teller takes the output batch of the previous teller's second mix and performs two further mixes, and so on through the full set of mix tellers.

At the end of this process, the batch of receipts will have undergone $2j$ re-encryptions and shuffles and are ready to be decrypted. Decryption will then be performed by a threshold set of decryption tellers who hold secret shares for the onion encryption.

Once all this is completed, the final, decrypted, anonymised votes appear in the final column of the WBB and these can be tallied in a conventional and universally verifiable fashion.

1.11 Auditing the Election

So far we have described the process under the assumption that all the steps are executed faithfully. We do not want the integrity of the election to rely on any of the entities involved behaving correctly and so we now introduce the mechanisms that serve to detect any malfunction or corruption.

1.11.1 Auditing the Ballot Generation Authority

The first place that things could go wrong is in the creation of the ballot forms. If a ballot form is incorrectly constructed, in the sense that the candidate list shown on the form does not correspond to the order given by the seed value buried in the onion value on the form, then the voter's choice will not be accurately encoded. We therefore need a mechanism to detect incorrectly constructed forms, without revealing keys, seed values or randomisation values.

If, as we have done above, we assume that the ballot forms are created in advance, we can perform a random audit on a proportion of the forms. So, we require the ballot creation authority (or authorities) to create an excess number of forms, perhaps twice as many as actually required, and allow independent organisations to make random selections of an appropriate proportion. For example, we might allocate five auditing organisations and allow each to select up to 10%. For these selected forms, the seed and/or randomisation factors are revealed, so allowing the auditors to recompute the values and confirm that they agree with those printed on the forms.

A rather elegant way of revealing the audit information for selected forms whilst ensuring that it is kept secret for ballot forms that are used to cast votes, is the "Scratch and Vote" mechanism of Adida and Rivest, [3]. Here, the audit information is printed on the ballot forms but concealed by a scratch strip. Revealing the information by removing the strip automatically invalidates the form for voting. Previously, this information was revealed by having the decryption tellers on-line to extract the seeds for audited forms. The Adida/Rivest approach avoids the need to have the tellers on-line.

The process of auditing a ballot form is accomplished by recomputing the values on the form from the cryptographic values. Thus the onion is recomputed from the seed value, randomisation and the teller public keys. The candidate order is also recomputed using the publicly known function σ from the seed space to space of permutations of the candidate set. If these agree with the values printed on the form, then the voter may conclude that the form was correctly formed by the device.

In addition to the audits performed by appropriate authorities before, during and after the election, we can invite the voters also to choose forms at random for audit. We will discuss such possibilities in detail later.

1.11.2 Auditing the Mix Tellers

Next, we need to confirm that the mix tellers perform all their actions correctly, i.e: each column is a re-encryption and permutation of the previous column, without revealing the permutations. Numerous techniques have been proposed to achieve this, for example the approach proposed by Neff, [28]. For comprehensive descriptions of various techniques see [3] or [12]. The technique that we describe here, as it is both rather intuitive and flexible, is that of *randomised partial checking* [21]. Half the links are randomly chosen to be revealed and verified. The choice of links, whilst essentially random, is carefully constrained in such a way as to ensure that no decrypted vote can be traced back to the original ballot receipt.

The selection process is best described as follows. Recall that each mix teller performs two mixes and so has three columns of the WBB associated with it: an input column, a middle column (the output of its first mix) and a final output column (the output of its second mix, which also serves as the input for the next teller). The auditor goes down the middle column and for each term makes a random *left/right* choice. If *left* is chosen, the teller is required to reveal the incoming link (the source term in the previous column) and the associated re-randomisation factor for that link. Similarly, if *right* is chosen, the teller must reveal the outgoing link. Thus each link has a 50/50 chance of being audited and yet our construction ensures no complete link across the two mixes associated with the teller.

The figure 1.4 illustrates this. The first column, marked *posting 1*, is the input to the first teller. The second column is the output of its first mix and the third column is both the output of the second mix and the input to the first mix of the next teller. The auditor makes random L/R selections down this middle column and the teller must reveal the incoming or outgoing link information accordingly. Similarly for the fourth column, that is the middle column for the second teller.

For each of these audited links, the teller must reveal appropriate audit information, for example the re-randomisation fact or a ZK proof of re-randomisation. The auditors confirm that the target term is indeed a genuine re-encryption of the claimed source term. These calculations may be confirmed by anyone. The process is repeated with independent auditing selections for each of the tellers.

1.11.3 Auditing the Decryption Tellers

Finally, we also need to confirm that the final decryptions are performed correctly. Here we can be more direct and can audit every decryption step as we do not need to worry about anonymity at this stage as this is already provided by the mix phase. Thus we need simply check that each decrypted vote

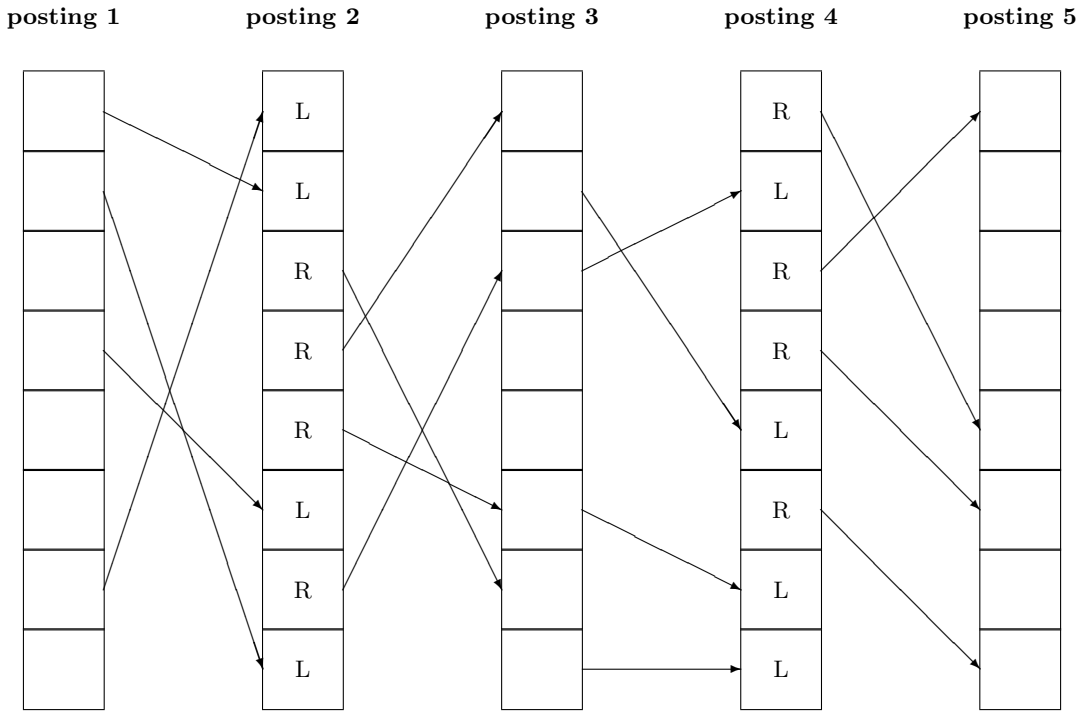


Fig. 1.4. Revealed links for auditing of two tellers

does indeed encrypt to the corresponding term in the previous column. Given that we are using randomising encryptions here, the process of checking the correctness of the decryptions is not quite trivial: we cannot simply perform the encryption of the claimed plaintext and check the result agrees with the ciphertext, and of course we don't want to reveal the secret keys. Here we can use the ZK proof techniques of Damgård *et al*, [11], to check that all partial decryption shares are valid and the final combination of shares is publically verifiable.

1.12 Threats and Trust Models

We now discuss a number of known threats against voting systems, in particular voter-verifiable schemes. For some of these, the scheme described above is already robust. For others, such as ensuring the secrecy of the information created by the ballot generation authority, chain of custody etc, we will need to introduce enhancements, described in the following section.

1.12.1 Leaky Ballot Creation Authority

So far, we have assumed that a single authority is responsible for the generation and printing of the ballot forms and proposed random pre-auditing to ensure that ballot forms are correctly constructed and are suitably randomised. Whilst this removes the need to trust the authority with respect to the integrity requirement, we still need to trust the authority to preserve the secrecy of the seed value and the onion/candidate list association. Clearly, if the ballot authority were to leak this information, the scheme would become susceptible to coercion or vote buying.

1.12.2 Chain of Custody

Here the threat is to the integrity and secrecy of the ballots forms between the time of their creation and use. We need to provide mechanisms to ensure that none of the onion/candidate list associations are leaked during storage and distribution. We also need to guard against corrupt ballot forms being smuggled into the system after the random auditing phase.

Various counter-measures are possible, for example, anti-counterfeiting measures could be used to prevent fakes being introduced. Ballot forms could be kept in sealed envelopes to be revealed only by the voters in the booth. Alternatively, a scratch card style mechanism along the lines suggested in [37] could be used to conceal the onion value until it is revealed at the time of vote casting. The ballot forms would also need to be stored and distributed in locked, sealed boxes. Further random audits could be performed throughout the voting period, perhaps giving voters the opportunity to audit forms too.

All of these counter-measures are rather procedural in nature and so require various trust assumptions. We will see shortly how on-demand printing and distributed construction of encrypted ballots along with post-auditing provide arguably stronger counter-measures, i.e. requiring weaker trust assumptions.

1.12.3 Chain Voting

Conventional pencil and paper elections are vulnerable to a style of vote buying known as chain voting. The UK system in particular is vulnerable. Here, the ballot forms are a controlled resource: on entering the polling station, the voter is registered and marked off on the electoral roll. They are given a ballot form which they take to the booth, mark and then cast in the ballot box. In principle, officials should observe the voters casting their ballot.

The attack works as follows: the coercer smuggles a blank ballot form out of the polling station. The controls on the distribution of the forms should

make this a little tricky, but in practise there are many ways it could be achieved by for example bribing an official or sleight of hand. Having marked the form for the candidate of their choice, the coercer intercepts a voter as they enter the polling station. The voter is told that if, when they exit the polling station, they hand a fresh, blank form back to the coercer they will receive a reward. The coercer can now use this form on the next victim and so on. Note that, once the attack is initialised, the controls on the ballot forms works in the coercer's favour: if the voter emerges from the polling station with a blank form, it is a strong indication that they did indeed cast the marked form they were given by the coercer.

For conventional, paper ballot systems, there are known counters to this threat, [22]. However, for voter-verifiable schemes with preprinted ballot forms the problem rears its head with a vengeance as the coercer who has obtained prior sight of Prêt à Voter ballot form is able to check via the WBB that the ballot has been cast as required.

1.12.4 Side Channels and Subliminal Channels

A common concern with electronic voting is that the device that captures the voter's choice may somehow leak this information, over a hidden wire, wifi, electromagnetic radiation etc. Recently a model of Nedap touch screen machine was decertified in the Netherlands precisely because it was found that votes could be detected several metres away by measuring EM radiation. All touch screen machines and most cryptographic schemes are potentially liable to such side-channel attacks.

More subtly, the device might exploit some form of subliminal channel to encode information in a legitimate channel, for example the WBB. Karlof et al, [25], describe the possibility of semantic and random channels in the original Chaum scheme and a version of the VoteHere scheme.

Most electronic voting systems necessarily involve the voter's choice being communicated to a device that records or encrypts the vote and so are potentially vulnerable. A important feature of Prêt à Voter is that the voter does not need to communicate their choice to any device. As remarked earlier, the vote is encoded in a randomised frame of reference. All that is encrypted is the information that defines the frame and this encryption and this can be performed in advance, without any knowledge of the vote value or indeed the identity of the voter who will eventually use the ballot form. This means that side-channel and subliminal channel attacks are neatly side-stepped: the device simply does not learn the information and so cannot leak it even if it had access to suitable channels.

1.12.5 Kleptographic Channels

Whilst Prêt à Voter avoids side-channels and subliminal channels, a further subtle vulnerability can occur where a single entity is responsible for creating cryptographic variables. So called *kleptographic* channels as described in [40] are a possibility in Prêt à Voter if we use a single authority for the generation of the ballot forms. The possible relevance of such attacks to cryptographic voting schemes is described in [27]. The idea is that the entity may carefully choose the values of the crypto variables in order to leak information to a colluding party.

In the case of Prêt à Voter, the Authority might choose the seed values in such a way that a keyed cryptographic hash of the onion value indicates the candidate order. The keyed hash would be shared with a colluding party. Clearly, executing such an attack would require quite a bit of searching and computation to find suitable seed values. Such an attack could pass unnoticed: the distribution of seed values would look perfectly random to anyone ignorant of the hash function. We will see shortly how introducing a pseudo-random generation of seeds or a distributed construction for the ballot forms in which a number of entities contribute to the entropy of the crypto variables counters this style of attack.

1.12.6 Retention of the Candidate List

In order to avoid coercion threats, it is essential to ensure that the association between the receipt and the LH strip, that carries the candidate list, is eliminated. Otherwise, a coerced voter may be induced to smuggle the LH portion of the ballot form out of the polling station in order to show the coercer how she voted. Various measures are possible, for example, a mechanical device that enforces the destruction of the LH strip. Perhaps the neatest approach is to ensure an ample supply of *decoy* LH strips so that a voter who is being coerced could simply pick up an alternative strip showing a candidate order that meets the coercer's requirements. A further simple technique is to segment the LH strip horizontally between the candidate names in such a way as to ensure that the strip will fall apart once detached from the RH strip.

1.12.7 Collusion between Mix Tellers and Auditors

Collusion between the mix tellers and the auditors can undermine integrity. If a teller knows in advance which links will be audited, they can corrupt ballots along the unaudited links with impunity. Drawing the tellers and auditors from hostile organisations helps reduce the likelihood of collusion. A publicly verifiable mechanism to generate the selections based for example on a public

lottery is another possibility. A more technical approach, based on the Fiat and Shamir heuristic [17] is to derive the audit selections from a pre-determined cryptographic hash of the posted information.

If re-encryption mixes are used, then is it feasible to re-run the mixes, or indeed run several mixes in parallel using independent tellers and independently audit these using different auditing authorities. The fact that mixes and audits can be rerun without undermining privacy is one of the key advantages of using re-encryption mixes over decryption mixes. With decryption mixes, the set of terms generated at each stage will be the same each time the mix is rerun and so it is not possible to independently audit different runs of the mixes without revealing too much information.

1.12.8 Ballot Stuffing

Another danger, common to all voting systems, is that of voting officials smuggling in extra ballots into the count. Voters checking their receipts does not detect ballot stuffing. One simple counter is simply to check that the number of votes counted matches the number registered as having been cast. A more sophisticated approach is to post the names on the WBB of voters who have participated, so that anyone listed who did not participate can object. This counter-mechanism has dangers of its own, e.g. forced abstention attacks: voters are told that if they vote as all they will be punished. The balance between such threats is difficult to evaluate and probably depends on context.

1.13 Enhancements and Counter-measures

For many of the threats that we have identified, Prêt à Voter is either resistant or we have suggested counter-measures. To address the remaining threats we now describe some enhancements to the basic scheme.

1.13.1 On-demand Generation of Prêt à Voter Ballot Forms

Generating the ballot forms in advance has several; attractive features: simplicity and the ability to catch problems early. It does however have the disadvantage that we have to worry about chain of custody issues. In this section we present a mechanism to generate the forms *ab initio*, including generating the seed entropy, in the booth. In the next section we will describe a distributed construction of encrypted ballot forms.

We assume that the public key of the tellers, $PK_T = (g, n)$, is suitably certified and publicised and that there is a publicly agreed function, σ, from

the seed space in to the set of permutations of the candidates. For the purposes of this section we will further assume that each booth device creates two public key pairs PK_{b1}, SK_{b1} and PK_{b2}, SK_{b2}.

Anne is now given a form at random that carries only a unique, random serial number ξ. When she enters the booth, she feeds this form into a device that reads the serial number. It applies its first secret, signing key SK_{b1} to ξ to generate the seed value ρ from which it computes the candidate order π. It also computes the onion value θ by encrypting the seed value, ρ, with the public, threshold teller key PK_T. The randomisation used in this encryption is generated by signing ξ with its second secret key SK_{b2}.

It prints the candidates on the LH column of the form in the appropriate order. On the RH strip it prints the onion value to give a conventional Prêt à Voter form. To facilitate auditing should the form be selected for audit, we can use a variant of the Adida/Rivest off-line audit mechanism, [2]: the device prints information to enable checking the well-formedness of ballot forms selected for audit. We will discuss this in more detail later, for the moment we denote the audit information by μ. For a form used for voting, this information will be discarded along with the candidate order, but it will be preserved on a form destined for audit.

Thus, the seed is computed as:

$$\rho := \{\xi\}_{SK_{b1}} \pmod{n}$$

the randomisation factor as:

$$\zeta := \{\xi\}_{SK_{b2}} \pmod{n}$$

the candidate order as:

$$\pi := \sigma(\rho)$$

and finally the onion value:

$$\theta := \{\rho, \zeta\}_{PK_T}$$

The figures illustrate this: Figure 1.5 shows a typical proto-ballot form bearing only a serial number. Figure 1.6 shows the form after the booth device has computed and printed the the candidates and audit information μ on the LH side, and the onion value at the bottom of the RH side.

The generation of the seed and randomisation as deterministic signatures on the serial number serves to counter kleptographic attacks in which the

	ξ

Fig. 1.5. Proto-ballot form with serial number

Idefix	
Asterix	
Obelix	
Panoramix	
μ	ξ
	Θ

Fig. 1.6. Prêt à Voter ballot form showing the onion and audit information

device leaks information over subliminal channels by careful selection of the seed value, [27]. The proposed mechanism ensures that the device has no freedom in the choice of the seeds.

Later we will discuss cut-and-choose mechanisms to partition ballot forms into ones for audit and ones for casting votes.

1.13.2 Distributed generation of Paillier Encrypted Ballot Forms

The disadvantage of the approach above is that the booth device necessarily learns the association of the candidate order and onion value. We could implement these devices in such a way as to ensure that all information is erased once the form is printed. However, guaranteeing this may be difficult and, furthermore, there are dangers of the information being leaked over side-channels. The construction of this section seeks to address these issues.

The onions are be generated by a set of l clerks in such a way that each contributes to the entropy of the cryptographic values from which the candidate list is derived. Furthermore, these values remain encrypted throughout. As a result, all the clerks would have to collude to determine the seeds values.

As before, we assume a set of decryption tellers who hold the key shares for a threshold Paillier algorithm with Teller public key PK_T: (g, n). This public key is known to the Clerks and is used in the construction of the encrypted ballot forms.

The first clerk C_0 generates a batch of initial seeds $\bar{s}_i^0$ drawn at random from Z_n^*. From these, C_0 generates a batch of initial onions by Paillier encrypting each $\bar{s}_i^0$ under the Teller key:

$$\mathcal{E}(\bar{s}_i^0, x_i^0) = (g^{\bar{s}_i^0} \cdot (x_i^0)^n) \pmod{n^2}$$

for fresh random values $\bar{x}_i^0$ drawn from Z_n^*.

The remaining $l-1$ Clerks now perform re-encryption mixes and transformations on this batch of onions: each Clerk takes the batch of onions output by the previous Clerk and performs a combined re-encryption along with an injection of fresh seed entropy into the seed values. For each onion, the seed entropy is drawn from Z_n^* is injected into the seed value of the onion. The seed entropy and randomising factors will be independently chosen for each onion.

More precisely, for ith onion of the $j-1$th batch $\Theta_i^{j-1} := \mathcal{E}(s_i^{j-1}, x_i^{j-1})$, the jth Clerk C_j generates fresh, random values $\bar{x}_i^j$ and $\bar{s}_i^j \in Z_n^*$ and multiplies Θ_i^{j-1} by $\mathcal{E}(\bar{s}_i^j, \bar{x}_i^j)$:

$$\Theta_i^j = \mathcal{E}(s_i^j, x_i^j) = \mathcal{E}(s_i^{j-1}, x_i^{j-1}) \cdot \mathcal{E}(\bar{x}_i^j, \bar{x}_i^j) = \mathcal{E}(s_i^{j-1} + \bar{s}_i^j, x_i^{j-1} \times \bar{x}_i^j)$$

where

$$\begin{aligned} x_i^j &:= x_i^{j-1} \times \bar{x}_i^j \pmod{n^2} \\ s_i^j &:= s_i^{j-1} + \bar{s}_i^j \pmod{n} \end{aligned}$$

Having transformed each onion in this way, the Clerk C_j then performs a secret shuffle on the batch and outputs the result to the next Clerk, C_{j+1}.

So the final output after $l-1$ mixes is a batch of onions of the form: $\Theta_i := \mathcal{E}(s_i, x_i) = (\alpha^{s_i} \cdot (x_i)^n)$ where:

$$x_i = x_i^l \textit{ and } s_i = s_i^l$$

$$s_i = \sum_{j=0}^{l} \bar{s}_i^j \pmod{n}, \qquad x_i = \prod_{j=0}^{l} \bar{x}_i^j \pmod{n^2}$$

As the seed values, and hence the candidate orders, remain encrypted, none of clerks knows the final seed values and they would all have to collude to determine them. These onions can now be stored and distributed in this form, thus avoiding the chain of custody problems mentioned above. Kleptographic channels are also avoided as no single entity is able to choose the seed values in such a way as to leak information.

1.14 Transforming Receipt Onions into Booth Onions

We could now proceed to use these onions, which we will refer to as *receipt onions*, directly in the construction of the ballot forms, much as described in the previous section. However, we would like to ensure that the booth device does not learn the value of the onion used in the receipt in order to avoid a single device knowing the association of receipt onion and candidate order. To this end we describe a further stage in the construction to create re-encryptions of these onions that will be available to and decryptable by the booth device.

We introduce a number of *re-encryption clerks* and we assume that the onions created in the previous phase described above have been printed on *proto*-ballot forms. That is, each ballot form will have printed on it a unique onion drawn from the set generated as described above. The first of these re-encryption clerks takes the batch of such forms and for each performs a re-encryption of the onion on the form. It then covers the onion with a scratch strip and overprint this with the re-encrypted onion value. The resulting batch of forms, now showing only the re-encrypted onion values, is shuffled and passed on to the next re-encryption clerk who repeats the process. This can be repeated for as many re-encryptions as we require, depending on how high we want the collusion threshold to be placed. Just one such re-encryption might be judged sufficient in some cases.

The upshot of all this is that we have a batch of proto-ballot forms on each of which is visible only the final (multiply) re-encrypted onion value overprinted on a (multi-layer) scratch strip. We refer to these uppermost onions as the *booth onions*. Under these scratch strips are the *receipt onions*. The voting procedure is now as follows: again Anne picks up a form at random and proceeds to the booth. The scratch strip should be kept intact until the point at which the ballot is cast. In the booth, the device reads the visible, booth onion, decrypts it and computes the candidate list. The device prints the candidate list and audit information on the LH side of the form resulting in a conventional Prêt à Voter ballot form. Anne fills in her selection, removes and discards the LH column and leaves the booth. In the presence of an official, the scratch strip is now removed to reveal the receipt onion (and in the process destroying the *booth* onion). Vote casting completes as before. Note that the use of the scratch strips to conceal the receipt onions also helps counter chain voting and chain of custody attacks.

An issue that we need to deal with in the above construction, is how the booth device is to perform the decryption of the onion. A possibility is to have the tellers available online and have the booth device communicate the onions value to them to perform partial decryption of onions. We could arrange for the booth device to hold a share of the key and so perform the final step of the decryption. Ideally we would like to transform the receipt onions, that

currently can only be decrypted by a threshold set of the tellers, to a form that can be decrypted unaided by the booth device, without requiring any tellers to be available on-line during the voting phase.

To this end, we introduce a further transformation phase in which an $t-1$ subset of the tellers perform partial decryptions with their share of the teller key, but stopping short of full decryption (which will be performed at the last moment by the booth device). Here we describe how this can be accomplished for ElGamal ciphertexts. A similar, but rather more elaborate construction can be provided for Paillier terms, but we omit the details due to space constraints.

We assume a Shamir secret sharing scheme with threshold of size t such that the secret key K is given by $K = f(0)$, where f is a randomly chosen polynomial of degree $t-1$ and $k_i = f(i)$ is the share allocated to the i'th teller. For a set of tellers, with index set S, of size t the secret is given by $K = f(0) = \sum_{i \in S} \mu_i^S \cdot f(i) = \sum_{i \in S} \mu_i^S \cdot k_i$, where the μ terms are the appropriate Legendre coefficients for the set S. The tth element of this set will the booth device.

We progressively decrypt an ElGamal ciphertext $(x_0; y_0)$ by progressively factoring out the $(x_o)^{\mu_i^S \cdot k_i}$ shares from the second y_0 term of the ElGamal ciphertext pair. For $i \in \{1, ...t-1\}$, the i'th teller performs the following partial decryption on the $i-1$'th ElGamal term:

$$\mathcal{D}_i(x_0;\ y_{i-1}) = (x_0, y_i) = (x_0;\ y_{i-1} \cdot (x_0)^{-\mu_i \cdot k_i})$$

We stop short of performing the final, t'th booth decryption so obtaining the booth onion. The final decryption step will be performed on-demand by the booth device using its share of the key:

$$\mathcal{D}_t(x_0;\ y_{t-1}) = \rho_i = y_{t-1} \cdot (x_0)^{-\mu_t \cdot k_t}$$

1.15 Auditing "on-demand" Ballot Forms

The mechanisms described above allow for the on-demand printing of ballot forms in the booth. This has advantages in terms of removing the need to trust a single entity to keep the ballot form information secret and avoids chain of custody issues. On the other hand, it means that we can no longer use pre-auditing of pre-printed ballot forms. Given that we want to avoid having to trust the device in the booth, we must introduce alternative techniques to detect and deter any corruption or malfunction in the creation of the ballot forms. To address this we introduce a cut-and-choose element into the protocol.

To this end, voters could be furnished with two or perhaps more encrypted ballot forms. For each of these, the booth device decrypts the onion, computes the candidate order and prints this on the form. The voter selects one at random to cast her vote: the others will be audited and discarded. Care must be taken to avoid introducing dangers of double voting or chain voting etc. Double voting is probably fairly easily countered by the supervised casting of ballot receipts in the presence of officials who ensure that only one form is cast and the voter's name is marked as having voted.

Chain voting threats might be a little more delicate to counter: here a malicious voter, Yves, secretes a decrypted ballot form and smuggles it out of the polling station. Yves marks this form with the candidate of his choice and passes it to another voter who is required to cast their vote using this form. Yves can subsequently check whether the voter complied by checking the WBB. Keeping a log of serial numbers issued to a voter, in the manner of known counters to chain voting [23], would help here. If we employ the scratch strip technique of Section 1.13.2, this has the effect of concealing the receipt onion value and so thwarting chain voting attacks. The receipt onion would only be revealed at the time of casting the ballot and verified in the presence of officials.

The double sided forms of [31] provide a mechanism to keep a clear account of the distribution of ballot forms. Here, each side of a form carries an independent onion. For each side, the onion is decrypted and the candidate order printed. This results in two independent Prêt à Voter ballot forms being printed, one on each side of the form. The voter arbitrarily selects one side to vote and the other for audit. The forms actually have a third, blank column opposite the candidate list on the other side, as shown in figures 1.7 and 1.8. Thus, detaching the candidate list on the voted side detaches the blank column of the flip side, so leaving an intact form for audit. The two sides of the resulting receipt where the voter has cast a vote for Asterix on the second side is shown in figures 1.9 and 1.10.

Obelix		
Asterix		
Idefix		
Panoramix		
499052712	$7rJ94K$	

Fig. 1.7. Prêt à Voter ballot form; side 1

Obelix		
Asterix		
Idefix		
Panoramix		
499052712	$Yu78gf$	

Fig. 1.8. Dual Prêt à Voter ballot form; side 2

Obelix	
Asterix	
Idefix	
Panoramix	
499052712	$7rJ94K$

Fig. 1.9. Prêt à Voter ballot receipt; auditable side

X	
$Yu78gf$	

Fig. 1.10. Dual Prêt à Voter receipt; vote carrying side

1.16 Checking the Construction of the Ballot Forms

Using Paillier encryption suggests alternative techniques to demonstrate correctness of the construction of audited forms. If we use the on-demand generation of seeds and randomisation described in Section 1.13.1, the booth device will know randomisation and can either reveal it for audit purposes or to construct a ZK proof. Alternatively, if we use pre-prepared onions, the booth device will not immediately know the randomisation. However, Damgård et al, [11], show that for Paillier, knowledge of the secret key allows recovery of the randomisation. This rather remarkable fact about Paillier encryption seems to be surprisingly little known. The Damgård et al paper actually gives the results for their generalisation of Paillier to modulo n^{s+1}, but for simplicity of presentation we give the result for Paillier's original modulo n^2 version of the algorithm.

Suppose that the decryption teller P is given a ciphertext $c = g^m \cdot r^n \pmod{n^2}$. He recovers the plaintext m from which he can compute $g^{-m} \pmod{n^2}$ and can extract can extract randomisation term: $r^n = c \cdot g^{-m} \pmod{n^2}$. Now P needs to take the n'th root of this term. He computes a value a such that $a \cdot \lambda + 1 = 0 \pmod{n}$ and then computes:

$$(r^n)^{\frac{a\cdot\lambda+1}{n}} \pmod n = r^{a\cdot\lambda+1} \pmod n = r \pmod n$$

We observe that to extract the randomisation factor does not in fact require knowledge of λ but only of the value $\eta := (a \cdot \lambda + 1)/n$. Furthermore, knowledge of η does not entail knowledge of λ. As a result we can safely allow the official auditing devices to know η to enable them to extract the randomisation factor and use this along with the seed value revealed by the booth device to check the well-formedness of the audit form.

A few remarks are in order here. Firstly, if the threshold schemes was set up by a dealer who distributed the shares to the tellers, then the dealer can compute η. However, we may prefer to avoid having a dealer know λ and so would use a distributed scheme to set up the shares in such a way as no single entity knows λ. In this case it is not so clear how η can be computed. This will be investigated in a forthcoming paper.

Another issue is that printing the audit information on the LH column of the forms may undermine the decoy strip mechanism by creating a verifiable link between the LH and RH columns. If we arrange for the randomisation factor to be only recoverable by the official auditing devices at the time of casting, then this lessens the danger of links being established between the LH and RH strips of ballot forms used to cast votes. In fact, we could arrange for the booth device to print an encryption of the seed value under a public key for the official audit devices. The official audit devices would then be required to print the revealed seed and randomisation value on audit forms so that the checks can be independently verified.

Ballot forms could be audited at several points in the process. A first check would be performed at the time of casting the receipt: official auditing devices would be available at the registration desk. If we have used the distributed construction of receipt and booth onions, then for any given form, the booth and receipt onions should have the same seed value but different randomisations. Thus the official audit device is able to directly audit the correctness of the receipt onion w.r.t. the candidate order shown using the seed value revealed by the booth and the randomisation recovered using knowledge of η.

As we don't want to trust the official audit devices we introduce *voter helper organisations*, [3]: representatives of the various parties etc. provide auditing devices at the polling stations so that further, independent checks can be performed just after casting their receipt. These audit devices could also check the validity of the digital signatures applied at the time of casting.

Additionally, auditing could take place on material posted to the WBB. All the information on both sides of the receipts would be posted. The voted and auditable sides should be posted to separate regions of the WBB in such a way as to loose any association between the two sides. The voted sides would be processed via the tabulation mixes whilst the auditable sides are available

to be verified by anyone. Note that, in principle, anyone can write an auditing program: the algorithms are all public and indeed quite standard.

Audit forms posted to the WBB could themselves be decrypted by the tellers in a sort of tabulation mix, except that here we would carry the candidate order through the mix and check that the order computed on decryption of seed matches. This strategy avoids any need to reveal audit information and it might be tempting to adopt it as the sole ballot auditing mechanism. In practice, it is probably preferable to use it as a supplementary mechanism to provide added assurance, as corruption would be detected very late in the process. Audit forms could be required to carry booth identifiers so that any corruption detected can be traced back to the offending booth device and isolated.

1.17 Re-encryption/Tabulation Mixes

Paillier encrypted terms can be put through a conventional re-encryption mix, but Prêt à Voter ballot receipts do not have the form of straight Paillier encrypted terms, so they cannot be treated quite so straightforwardly. In addition to the onion term we have the index value, in the clear as it were. An obvious approach would be to send the receipt terms through the mix re-encrypting the onions whilst leaving the index values unchanged. The problem with this is that an adversary is able to partition the mix according to the index values. There may be situations in which this is acceptable, for example large elections in which the number of voters vastly exceeds the number of voting options. In general this seems unsatisfactory.

A more satisfactory solution, at least for the case of a simple selection of one candidate from the list, is described here. In this case we restrict ourselves to simple cyclic shifts from the base ordering of the candidates. For single candidate choice voting this is sufficient to ensure that the receipts do not reveal the voter's selection. For more general styles of election, for example in which voters are required to indicate a ranking of the candidates, we of course need to allow full permutations of the candidate list. We will discuss how to achieve full mixing in the more general case in Section 1.18 .

Suppose that s is the seed for the a given ballot form and ν is the number of candidates, then we can simply let $s \pmod{\nu}$ be the shift of the candidate list. We can absorb the index value ι on the ballot receipt into the Pailler onion:

$$\{\iota, (\alpha^{-s}.y^n)\} \rightarrow (\alpha^{\iota}.\alpha^{-s}.y^n) = (\alpha^{\iota-s}.y^n)$$

Note that, for convenience, we encrypt $-s$ rather than s. This gives a pure Paillier encryption of the value $\iota - s$ which, taken modulo ν, gives the voter's original candidate choice in the canonical base ordering. These pure Paillier

terms can now be sent through a conventional re-encryption mix as described earlier.

1.18 Handling Full Permutations and STV Style Elections

Handling full permutations of the candidate list is straightforward in the 2005 version of Prêt à Voterthat uses RSA onions and decryption mixes [8]. The permutation of the candidates shown on the original ballot form is built up as the product of permutations derived from seed values buried in each layer of the onion. During the tabulation mix, as we reveal a new seed value at each stage of the mix, we simply apply the inverse permutation derived from this seed. As long as we arrange for these inverse permutations to be applied in the reverse order we undo the effect of the permutation applied in the construction of the ballot form. The overall effect is to output the indices (or rank vectors) in the canonical, base ordering, [8].

In order to deal with full permutations of the candidate list where randomising encryptions and re-encryption mixes are used, it is not immediately clear how to generalise the approach of Section 1.17. One solution is simply to have one onion against each candidate, encrypting the corresponding candidate code. For a single candidate selection, the ballot receipt would in effect simply be the onion value against the chosen candidate. This feels rather inelegant and inefficient in that it multiplies the number of onions required.

A rather neat way to deal with ranked, STV style, ballots, due to Heather [20], works as follows. Ballot receipts are formed as vectors of pairs comprising a rank value and an onion encrypting a candidate index. These receipts are posted as usual and can be checked as before. Before mixing starts, we introduce an initial normalising step in which each ballot vector is re-ordered into rank order. Once this is done, the ranking can be dropped and the ballots are sent through re-encryption mixes as tuples of onions.

Thus, for example, a ballot vector:

$$\{(3, \Theta_1), (1, \Theta_2), (4, \Theta_3), (2, \Theta_4)\}$$

would be normalised to:

$$\{(1, \Theta_2), (2, \Theta_4), (3, \Theta_1), (4, \Theta_3)\}$$

and then to:

$$\{(\Theta_2), (\Theta_4), (\Theta_1), (\Theta_3)\}$$

These ballot vectors are now fed into a sequence of re-encryption mixes in which each onion is independently re-encrypted but the the vector composition is preserved. Thus, the tuple above would be transformed to:

$$\{(\Theta'_2), (\Theta'_4), (\Theta'_1), (\Theta'_3)\}$$

Where the prime indicates re-encryption and Θ_i denotes an onion encrypting the index of candidate i. Once the ballots have gone through an appropriate number of re-encryption mixes, we move to the decryption phase. Now we can adopt a lazy decryption strategy: firstly we just decrypt the top, ranked first, onions for each vector and perform the first phase of the STV counting. When votes are transferred from eliminated candidates, the list of onions in a vector are rotated so that the second onion goes to the top of the list whilst the previously top vote is encrypted and drops down to the bottom of the list. Thus, the vector above might be partially decrypted to:

$$\{Candidate2, (\Theta_4), (\Theta_1), (\Theta_3)\}$$

and, on transfer transformed to:

$$\{(\Theta'_4), (\Theta'_1), (\Theta'_3), (\Theta'_2)\}$$

We only decrypt lower ranked onions as required by the counting, and typically we will need only to decrypt a few orders of the ranking before the counting process terminates. The transfer history of a vote is thus completely concealed, so countering "Italian" style attacks, in which a coercer requires a voter to use a certain identifying pattern in his low order rankings to prove how he voted. Where voters have not filled in all lower rankings, the remaining onions are simply inserted in the vector in the random order of the ballot form.

1.19 Remote Voting

The schemes that we have discussed so far are all supervised: voting takes place in a controlled environment such as a polling station and the casting of the vote occurs in the enforced isolation of a polling booth.

In remote systems, voters are able to cast their vote over a suitable channel, e.g: postal, internet, telephone, interactive TV etc. Here, the isolation of the voter whilst casting their vote cannot be enforced and the threats of vote

buying or coercion are consequently much more serious. It is sometimes argued that the difficulty in ensuring coercion-resistance is enough to render remote voting inadmissible.

To implement a scheme satisfying coercion-resistance, it is necessary to assume at least one step in the interaction that cannot be observed by the coercer, otherwise the voter and coercer would be indistinguishable to the system. Clarkson et al [10] have devised a remote implementation of Prêt à Voter with increased resistance to coercion, based on an approach of Juels at el [24]. Voters are issued with tokens that look like random streams but are actually encryptions of a "valid" string. Ordinarily, a voter casts their vote along with their token. If threatened by a coercer, they can corrupt the token in the knowledge that this will not be recorded as a valid vote. The voter can then recast their vote at some other time using their valid token. Valid tokens, and hence associated votes, are only identified during the tabulation phase, after anonymisation mixes. The coercer has no way to distinguish between valid and invalid vote.

Coming up with a way to assure voters that they have been provided with a valid token in such a way that this assurance cannot be transferred to a third party remains an open question, at least if we want to do this in a way that does not require vesting trust in some token distribution entity. Even if we suppose that a technical solution to this could be found, there remains that matter of public trust in such a mechanism. If voters do not believe in the mechanism they will again be open coercion.

1.20 Conclusions

Confidence in voting systems, particularly electronic systems, has been severely shaken in recent years by, most notably, the debacles in the 2000 and 2004 US presidential elections. It is rather troubling that the nation that proclaims itself the most technologically advanced and the torchbearer for democracy can make such an utter hash of voting technology. Many activists point to the evident deficiencies of many of the existing voting technologies and from this conclude that all voting technology must be deficient and go on to advocate a return to good old fashioned pencil and paper ballots along with hand counting. Whilst the critiques of most current voting technologies are certainly valid, to condemn all voting technology as a result is not a valid inference. And of course, pencil and paper is not immune to corruption.

In this chapter I have described an alternative response, based on systems providing transparency and auditability. Significant strides have been made of late towards voting systems that are trustworthy and yet sufficiently simple to be usable and to gain the confidence of the electorate. These *end-to-end* systems strive to avoid the need to place trust in devices, software, processes

or officials. Furthermore, the trustworthiness of the system ultimately rests with the electorate themselves.

I have presented the key ingredients of the Prêt à Voter scheme and its key properties. I have also described some of the known threats against such schemes and gone on to present some enhancements to the basic scheme that counter these threats. Arguably some of these threats are somewhat exotic and so the additional complexity of the enhanced versions might not be on balance be worthwhile, at least in some contexts.

There is doubtless scope for further innovations and simplifications of these schemes. More analysis of these schemes is required, and such analysis will have to be extended to a full systems-based approach taking account of the surrounding socio-technical system. Whilst such schemes appear to provide high levels of trustworthiness, at least in the eyes of experts, it is not clear that the public at large would understand and appreciate the rather subtle arguments and so be prepared to trust in them to the same extent as, say, familiar pen and paper systems. This leads us to some delicate socio-technical questions:

- To what extent can the properties of cryptographic systems be explained to the general public?
- To what extent is it necessary for the general public to understand in order to have sufficient trust in such systems?
- To what extent are the assurances of independent, impartial experts enough to engender trust?
- To what extent might it be necessary to compromise on trustworthiness in order to achieve understandability by, for example, replacing cryptographic mechanisms with simpler technology or processes? [31].

Somewhat paradoxically, the very transparency and auditability of the scheme may be an obstacle: the fact that errors and corruption can be detected, and in practice presumably errors will occur, may damage confidence. In conventional systems, most errors go undetected and, if detected, are handled quietly behind the scenes. This is a rather delicate issue, probably best addressed by those better qualified.

Another area that has seen very little exploration to date is that of effective recovery strategies. Verifiable schemes define mechanisms for detecting errors or corruption, but typically say little about how to respond when errors are detected. Clearly we do not want to abort an election if only a smattering of errors are detected, especially if these are negligible compared to the margin of the count. Where should the thresholds be placed at which recovery actions are be triggered and what are the appropriate response strategies?

I hope that this chapter has helped put the case for high assurance schemes like Prêt à Voter. Such schemes can provide far higher levels of assurance than traditional pen and paper and with much lower dependence on software, hardware, processes and officials and as such have the potential to help restore confidence on the processes of democracy.

1.21 Acknowledgements

The author would like to thank the members of the Newcastle Security Group, Ben Adida, Ran Canetti, David Chaum, Michael Clarkson, Bob Cunningham, Joshua Guttman, James Heather, Markus Jakobsson, Thea Peacock, Jean-Jacques Quisquater, Ron Rivest, Steve Schneider, Paul Syverson and Thomas Tjøstheim for helpful comments and discussions.

This chapter is dedicated to my late parents, to whom I owe everything.

References

1. Anonymity bibliography. http://freehaven.net/anonbib/.
2. B. Adida and R. L. Rivest. Scratch vote: Self-contained paper-based cryptographic voting. In *Workshop on Privacy in the Electronic Society, to appear*, 2006.
3. Ben Adida. Advances in cryptographic voting systems, MIT PhD thesis, July 2006.
4. O. Baudron, P.-A. Fouque, D. Pontecheval, G. Poupard, and J. Stern. Practical multi-candidate election system. In *Symposium on Principles of Distributed Computing*, pages 274–283. ACM, 2001.
5. J. Beneloh and D. Tuinstra. Receipt-free secret-ballot elections. In *Symposium on Theory of Computing*, pages 544 – 553. ACM, 1994.
6. D. Chaum. Untraceable mail, return addresses and digital pseudonyms. *Communications of the ACM*, 24(2):84–88, 1981.
7. D. Chaum. Secret-ballot receipts: True voter-verifiable elections. *IEEE Security and Privacy*, 2(1):38–47, January-February 2004.
8. D. Chaum, P.Y.A. Ryan, and S. Schneider. A practical, voter-verifiable election scheme. In *European Symposium on Research in Computer Security*, number 3679 in Lecture Notes in Computer Science. Springer-Verlag, 2005.
9. David Chaum and Torben P. Pedersen. Wallet databases with observers. In *CRYPTO '92: Proceedings of the 12th Annual International Cryptology Conference on Advances in Cryptology*, pages 89–105, London, UK, 1993. Springer-Verlag.
10. M. Clarkson and A. Myers. Coercion-resistant remote voting using decryption mixes. In *Workshop on Frontiers of Electronic Elections*, 2005.
11. I. Damgard, M. Jurik, and J. Nielsen. A generalization of Paillier's public-key system with applications to electronic voting, 2003.

12. George Danezis. Better anonymous communications, PhD University of Cambridge, July 2004. http://www.cl.cam.ac.uk/ gd216/thesis.pdf.
13. W. Diffie and M. E. Hellman. New directions in cryptography. In *Transactions on Information Theory, vol. IT-22, Nov. 1976*, pages 644–654. IEEE, 1976.
14. Y. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Transaction on Information Theory, Volume 31, 469-472*. IEEE, 1985.
15. A. J. Feldman et al. Security analysis of the Diebold Accuvote-TS voting machine. Princeton, Sept 2006. itpolicy.princeton.edu/voting/ts-paper.pdf.
16. K. Fisher et al. Puncscan, introduction and definition. www.cisa.umbc.edu/papers/FisherWote2006.pdf.
17. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology — Crypto '86*, pages 186–194, New York, 1987. Springer-Verlag.
18. Grumbaugh. Ancient greek gadgets and devices, 1966. New York, Crowell, ISBN 0837174279.
19. Andrew Gumbel. Steal this vote!, 2005. Nation Books.
20. J. Heather. Implementing STV securely in prêt à voter, 2007. To appear in Proceedings of Computer Security Foundations 20.
21. M. Jakobsson, A. Juels, and Ronald Rivest. Making mix nets robust for electronic voting by randomized partial checking. In *USENIX Security Symposium*, pages 339–353, 2002.
22. D. W. Jones. A brief illustrated history of voting, 2003. `http://www.cs.uiowa.edo/~jones/voting/pictures`.
23. D. W. Jones. Threats to voting systems, 2005. http://vote.nist.gov/threats/papers/threats_to_voting_systems.pdf.
24. A. Juels and M. Jakobsson. Coercion-resistant electronic elections, 2002.
25. C. Karlof, N. Sastry, and D. Wagner. Cryptographic voting protocols: A systems perspective. In *USENIX Security Symposium*, number 3444 in Lecture Notes in Computer Science, pages 186–200. Springer-Verlag, 2005.
26. T. Kohno, A. Stubblefield, A. D. Rubin, and D. S. Wallach. Analysis of an electronic voting system. In *Symposium on Security and Privacy*. IEEE, 2004.
27. M. Gogolewski et al. Kleptographic attacks on e-election schemes. In *International Conference on Emerging trends in Information and Communication Security*, 2006. http://www.nesc.ac.uk/talks/639/Day2/workshop-slides2.pdf.
28. A. Neff. A verifiable secret shuffle and its application to e-voting. In *Conference on Computer and Communications Security*, pages 116–125. ACM, 2001.
29. Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. *Lecture Notes in Computer Science*, 1592, 1999.
30. L. Adleman R. Rivest, A. Shamir. A method for obtaining digital signatures and public-key cryptosystems, 1978. Communications of the ACM, Vol. 21 (2), pp.120126.
31. B. Randell and P.Y.A. Ryan. Voting technologies and trust. *IEEE Security & Privacy*, November 2006.
32. R. L. Rivest. The three ballot voting system, 2006. theory.lcs.mit.edu/ rivest/Rivest-TheThreeBallotVotingSystem.pdf.
33. P.Y.A. Ryan. Towards a dependability case for the Chaum voting scheme, 2004. DIMACS Workshop on Electronic Voting – Theory and Practice.
34. P.Y.A. Ryan. A variant of the chaum voting scheme. Technical Report CS-TR-864, University of Newcastle upon Tyne, 2004.

35. P.Y.A. Ryan. A variant of the Chaum voting scheme. In *Proceedings of the Workshop on Issues in the Theory of Security*, pages 81–88. ACM, 2005.
36. P.Y.A. Ryan. Putting the human back in voting protocols. In *Fourteenth International Workshop on Security Protocols*, Lecture Notes in Computer Science. Springer-Verlag, 2006. To appear.
37. P.Y.A. Ryan and T. Peacock. Prêt à voter: a systems perspective. Technical Report CS-TR-929, University of Newcastle upon Tyne, 2005.
38. P.Y.A. Ryan and S. Schneider. Prêt à voter with re-encryption mixes. In *European Symposium on Research in Computer Security*, number 4189 in Lecture Notes in Computer Science. Springer-Verlag, 2006.
39. D. R. Stinson. Cryptography, theory and practice. In *Chapman and Hall, CRC*, 2006.
40. A. Young and M. Yung. The dark side of black-box cryptography, or: Should we trust capstone? In *Crypto'96*, Lecture Notes in Computer Science, pages 89–103. Springer-Verlag, 1996.