Simple Verifiable Elections

Josh Benaloh
Microsoft Research

June 14, 2006

Abstract

Much work has been done in recent decades to apply sophisticated cryptographic techniques to achieve strong end-to-end verifiability in election protocols. The properties of these protocols are much stronger than in any system in general use; however, the complexity of these systems has retarded their adoption. This paper describes a relatively simple but still effective approach to cryptographic elections. Although not as computationally efficient as previously proposed cryptographic approaches, the work presented herein is intended to be more accessible and therefore more suitable for comparison with other voting systems.

1 Introduction

Cryptographic protocols can allow each voter to ensure that *all* votes are counted properly, but they use sophisticated techniques that can be difficult for non-specialists to fully understand ([Chau81], [DLM82], [CoFi85], [Bena87], [PIK93], [BeTu94], [SaKi95], [CGS97], [BJR01], [FuSa01], [Neff01], [GZBJJ02], [JJR02], [Grot03], [Chau04], [Furu04], [Chau05], and [PBD05] offer just a sampling). Most cryptographic election systems can be divided into two phases

1. Each voter prepares and casts an encrypted ballot that represents the voter's intended selections. Once encrypted, these ballots — and even the identity of the voter that cast each one — can be made public.

2. Once all participating voters have cast their encrypted ballots, the set of encrypted ballots is cryptographically processed to produce a tally and a proof that the tally matches the set of ballots cast. In some cases, the original decrypted ballots are revealed, but the individual associations between the revealed ballots and the identities of the voters are removed.

Historically, the second or these two phases has received the most attention from researchers, and sophisticated cryptographic techniques have produced exceptionally strong results. The first phase went largely ignored until recently, and several new approaches have been built to support this verifiable encryption requirement.

While some systems have some overlap between the two phases described above, the work herein will provide a clear separation.

2 Background

While the intent of this work is to be, to the extent possible, fully self-contained, one well-established cryptographic tool will be taken as a primitive. *Threshold encryption* is a special form of public-key encryption. With ordinary public key encryption, one entity generates a public encryption key that can be used by others to encrypt values that can only be decrypted by the original entity. A wide-variety of public-key encryption technologies are available — with the most common being RSA encryption [RSA78].

Encryption is *not* used here to ensure election accuracy. Accuracy is instead verified by statistical means. The sole purpose of encryption is to protect voter privacy, and even a complete failure of the encryption system would not have any impact on the integrity of the election tally.

The reason for using threshold encryption instead of ordinary public-key encryption is to avoid any single point of potential privacy compromise. Threshold encryption does not allow decryption by any single entity. Instead, a pre-determined minimum number of *trustees* must work together to decrypt. Since we also don't want a small number of entities to be able to prevent the completion of an election, some robustness must be built in to allow for some failures among trustees.

With threshold encryption, participating entities each generate and publish an encryption key. When properly aggregated (in a publicly verifiable way), an aggregate public key is produced that allows others to encrypt data that can only be decrypted by the co-operative work of a predetermined number of the original key publishers.

Within the context of elections, each member of a large set of trustees — representing widely disparate interests — can generate and publish a public key. An aggregate of these keys can be formed in a public and verifiable manner, and this aggregate can be used by voters to encrypt their ballots. After the application of cryptographic operations which scramble but do not alter the semantic content of the set of encrypted ballots, any sufficiently large subset of the trustees can decrypt the final set to reveal the ballots and thereby determine the tally of the election.

The threshold number of trustees required to allow decryption should be set sufficiently high to prevent a small subset from working together to compromise privacy (note that even all of the trustees acting in concert cannot compromise the accuracy of the tally), but this number should not be set so high as to allow a small number of discontented trustees to prevent the completion of the election by refusing to co-operate with the final decryption. Depending upon the size of an election and political and practical constraints, reasonable threshold values for a particular election may, for example, be 3 of 5 trustees, 8 of 10 trustees, or 90 of 100 trustees.

One advantage of this approach is that it enables a clean mechanism for handling provisional ballots. With typical voting systems in current use, a preliminary tally of an election is produced while the legitimacy of provisional ballots is adjudicated. Any provisional ballot that is subsequently deemed valid is then added to the tally. This method undermines the privacy of legitimate voters who, for whatever reason, are forced to use provisional ballots. With the proposed cryptographic paradigm, provisional ballots can be included within any preliminary tally. Any provisional ballots later deemed to be ineligible could then be selectively removed by co-operation of a sufficiently large subset of election trustees. Thus, privacy is compromised only for *illegitimate* provisional ballots.

Numerous threshold encryption schemes exist (see, for example, [DeFr89], [Pede91], [GJKR96], [BoFr97], [Shou00], and [CDN01]) — many of which have a "probabilistic" character with many different encryptions possible for each value. A property available in many such threshold encryption schemes is the ability to transform an encrypted value to a different encryption of the same value without knowledge of the decrypted value. This "random re-encryption" capability will be used in the election system to be described. Random re-encryption is often *composible* in that if A is separately re-encrypted twice to form each of B and C, then the re-encryptions can be composed so that C can be regarded as a direct re-encryption of B without being linked or associated with A.

One threshold encryption scheme that offers this capability is described in the appendix, but it is important to stress that this approach does not depend on any one threshold encryption embodiment.

With threshold encryption available as a primitive, the vote casting and tallying phases can be described with a minimum of cryptographic details. For pedagogical reasons, the tallying phase will be described first in section 3 followed by a description of vote casting in section 4.

3 Ballot Tallying

The problem of verifiably tallying a set of ballots is easy if the privacy constraint is removed. Once the set of ballots — together with the identities of the voters that cast each one — is made public, voters can verify the accuracy of their own votes, and everyone can independently verify the accuracy of the tally. Privacy complicates the task, but cryptographers have spent years developing techniques for working with and proving things about encrypted data without revealing the original data. One such technique is known as an *interactive proof.*

3.1 Interactive Proofs

The notion of an interactive proof was developed more than twenty years ago and first published in [GMR85]. It's roots are found in challenge-response protocols which date back much further. The basic idea is that a *prover* can be repeatedly challenged by a *verifier* to answer queries that could only be answered if certain claims are true. A simple claim of this sort would be, "I have the decryption key that corresponds to a particular public encryption key." The verifier can select values, encrypt them with the public key, and challenge the prover to decrypt them. If the prover succeeds, the verifier should be convinced that the prover really does have the decryption key — without the prover having to reveal the actual key.

A somewhat more elaborate interactive proof can be used to show that two sets of encrypted ballots consist of exactly the same set of votes. Those familiar with the subject will recognize this interactive proof as virtually identical to the standard interactive proof that two graphs are isomorphic.

A set $\mathcal{B}$ of (threshold) encrypted ballots can be verifiably *shuffled* by an entity that does not know their decryptions as follows.

1. Each ballot $B_i \in \mathcal{B}$ is randomly re-encrypted to form a ballot B'_i which has exactly the same decryption as B_i.
2. The set of re-encrypted ballots $\{B'_1, B'_2, \ldots, B'_m\}$ is then randomly permuted to form a new ballot set $\mathcal{B}'$.[1]
3. A collection of n additional ballot sets $\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_n$ is generated — each in the same way as $\mathcal{B}'$ — by randomly re-encrypting each ballot and then permuting the set. [Think of $n \approx 100$.]
4. A set of n challenge bits $c_1, c_2, \ldots, c_n$ is generated.[2]
5. For each i such that $c_i = 0$, ballot set $\mathcal{B}_i$ is shown to be equivalent to ballot set $\mathcal{B}$ by revealing all of the re-encryption and permutation data used to create ballot set $\mathcal{B}_i$.
6. For each i such that $c_i = 1$, ballot set $\mathcal{B}_i$ is shown to be equivalent to ballot set $\mathcal{B}'$ by composing the re-encryption and permutation data used to create ballot set $\mathcal{B}_i$ with that used to create $\mathcal{B}'$ and then revealing the composition.

If any ballot set $\mathcal{B}_i$ can be shown to be equivalent to *both* $\mathcal{B}$ and $\mathcal{B}'$, then this proves with absolute certainty that $\mathcal{B}$ and $\mathcal{B}'$ are equivalent ballot sets. However, a direct demonstration of equivalence reveals which individual ballots in $\mathcal{B}$ and $\mathcal{B}'$ are equivalent and thereby defeats the entire purpose of the shuffle. Instead, with the interactive proof process above, no direct equivalences are ever produced.

[1]Instead of randomly permuting the elements of the new ballot set, the re-encrypted ballots can simply be sorted — numerically, alphabetically, or lexicographically — to obscure the association between the original encrypted ballots and the re-encrypted ballots.

[2]This will be elaborated upon shortly.

It is possible to "defeat" the interactive proof if and only if every one of the n challenge bits are known or guessed in advance by the prover. Any attempt to guess n random challenge bits will succeed with probability only 2^{-n}. How the challenges are selected is therefore crucial to the viability of this approach.

It is possible to engage trusted entities in a protocol to select challenge bits, but there's a better way.

3.2 The Fiat-Shamir Heuristic

The so-called Fiat-Shamir heuristic ([FiSh86]) can be used to generate challenge bits within the interactive proof process — without need for an external challenger. All of the ballot sets $\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_n$ can be combined as input to a *one-way hash* function to produce a set of n challenge bits $c_1, c_2, \ldots, c_n$ which are then used, as previously, to determine which equivalences among the ballot sets will be revealed. Any attempt to alter any of the ballot sets in response to the challenge bits produced will change the input to the one-way hash and cause an entirely different set of challenge bits to be produced.

If 2^n distinct ballot sets are run through this process, one could expect to find one such that non-equivalent ballot sets $\mathcal{B}$ and $\mathcal{B}'$ would appear to have been shown to be equivalent. Thus, n must be chosen to be large enough to preclude this attack, and choosing $n = 100$ is more than sufficient.

3.3 Completing the Shuffle

The shuffling process now becomes simple and sequential. The original encrypted ballots are collected into a ballot set, the names of the voters and/or other identifying information associated with each ballot are removed, and any interested parties are invited to individually shuffle the ballot set. Each shuffle is accompanied by a complete interactive proof that the resulting ballot set is equivalent to the ballot set prior to that shuffle. As long as the accompanying proof is valid, the shuffled set is then passed to the next shuffler. If the proof fails, the newly shuffled set is ignored and the ballot set prior to that last shuffle is passed to the next shuffler. By failing to produce a correct shuffle, a party does no damage to the process and only fails to add a level of indirection between the original ballot set and the final ballot set.

The entire process is public and verifiable.

Every step can be published and verified at a later time. Once the final shuffling is complete, the encrypted ballot set is decrypted (again in a verifiable fashion) by any sufficiently large subset of the pre-selected trustees who hold the decryption keys for the threshold encryption.

It is important to note that while catastrophic failure of the threshold encryption system could compromise privacy, it would not impact the accuracy of the tally. Accuracy is guaranteed statistically and by the one-way hash. Indeed the only drawback of the use of the Fiat-Shamir heuristic is that it introduces a dependence on a cryptographic process to achieve accuracy. The cryptographic assumption here is relatively weak, and even the very substantial recent attacks on cryptographic hash functions like MD5 and SHA-1 do not provide any known attacks to their use in a verifiable election process (although more modern one-way hash functions would certainly be preferred).

4 Ballot Casting

Now that a complete process for verifiably tallying a set of encrypted ballots is in place, it remains to determine how individual voters can create encrypted ballots to represent their voting selections. This does not seem terribly difficult at first glance, since voters could use systems they trust to perform their encryptions, but this approach could leave voters vulnerable to coercion. If voters are to use "standard" equipment supplied by election officials, then how are they to develop confidence that the encrypted ballots that they create represent their true intentions?

Several clever schemes have been devised to solve this problem using cryptographic bindings, physical apparatuses, and/or creative uses of commonplace products (see, for instance, [Chau04], [Neff04], [Chau05], and [CRS05]), but these systems all impose additional requirements on voters. With substantial rates of voter error even in well-designed traditional systems, it is undesirable to add new requirements whose purpose may not be clear to voters.

In order to better understand what is possible, it is worthwhile to step back and examine the requirements on a ballot encrypting device. Such a device can have a rich user interface much like a traditional electronic voting device. It must be able to process a voter's selections and create an encrypted ballot that accurately represents those selections. However, such a device need *not* know the identities of voters using the device; it need *not* authenticate that users even have the right to vote; it need *not* limit voters to a single use; it need *not* record the encrypted ballots it creates; it need *not* have any remote communication abilities; and it need *not* be involved in the casting of ballots.

With these ancillary requirements removed, it is possible to validate the accuracy of ballot encryptions *without* forcing every voter to participate in ballot validation activities. It is not difficult to imagine a simple, stand-alone device which offers a rich user interface but has no responsibilities beyond providing the voter with an encrypted ballot representing the voter's selections. This is not at all difficult to construct, but how then are voters to gain confidence that these devices are accurately creating encrypted ballots which reflect their intentions?

The key here is separating the vote creation and vote casting functions. By doing so, voters can be relieved of any requirement to take actions outside of today's norm, but those who desire to do so — whether voters, election officials, or outside observers — can audit and validate the accuracy of any votes produced by a vote creation device.

It should be noted here that while this separation of roles can help provide voter privacy, it does not *guarantee* privacy.[3] It is easy to imagine a malicious vote creation device retaining data about votes that could later be combined with externally obtained identification data to compromise privacy. However, privacy is *not* the primary purpose of the role separation. If the vote creation device does not know *in real-time* who is using it, then it is unable to distinguish, for instance, between an ordinary voter and an inspector who is using the device for auditing purposes. A malicious device that does not know when it is more likely to successfully "cheat" can do no better than to cheat at random, and this allows integrity to be provided by a random auditing process instead of requiring each event to be individually audited.

4.1 The Typical Voter View

The process for a voter can be as simple as for any voter using an electronic voting device today.

A typical voter would enter a polling station and approach a vote creation device — perhaps with-

[3]No voting system can ensure privacy. There is no way to prove, for instance, that a hidden camera or other covert device has not been surreptitiously added to the system.

out even checking in with poll workers.[4]The voter would then interact with the device, make selections, and receive an encrypted ballot representing those choices. A variety of media could be used to carry this encrypted ballot. One option would be a paper card with a magnetic stripe such as those commonly used on public transit systems. The encrypted ballot and/or a short cryptographic thumbprint thereof[5] could be printed on the card while the full encrypted ballot would be recorded on the magnetic stripe.

Once in possession of an encrypted ballot, the voter would proceed to a poll worker and sign-in, presenting whatever identification may be required. The voter would then swipe the magnetic-stripe card through a reader to record the encrypted ballot together with the voter's name. The voter can leave the poll site with the magnetic-stripe card and, if desired, later confirm that this encrypted ballot is properly associated with the voter's name on the published set of encrypted ballots which will subsequently be used to produce a verifiable tally.

4.2 Auditing View

The only observable difference between the vote creation device described above and a traditional electronic voting kiosk would be one or more additional options offered by the vote creation device at the conclusion of the process. While a voter can simply take the vote created by the device and cast it as described above, a voter, official, or observer would be offered at least one of several additional options.

4.2.1 Immediate Decryption

One option would be to have the vote immediately decrypted by the vote creation device. If, after the production of an encrypted ballot, the user of the device selects this option, the voting device would provide additional data that allows the user to later check that the encrypted ballot matches the voter's intentions.

If a ballot is verifiably decrypted by a device, the ballot must be marked as invalid for casting since this verifiable decryption would serve as a receipt that could subject the voter to the potential of coercion. Marking an encrypted ballot as invalid for casting may be most easily accomplished by adding a second component to each encrypted ballot. Once the encrypted ballot is produced and can no longer be altered by the vote creation device, the vote creation device would then (upon selection made by the voter) either digitally sign an attestation of the legitimacy of the encrypted ballot or provide a verifiable decryption of the ballot. Encrypted ballots accompanied by proper attestations could be cast, while others would not be deemed as allowable for casting.

The question presented to the voter may be as simple as, "Do you wish to cast this vote?" If the answer is "yes", then an attestation allowing the encrypted vote to be cast is added to the vote. If the answer is "no", a verifiable decryption is provided which the voter may keep for later verification or simply discard. The voter can then be given the option to change selections and, when complete, a new encrypted vote will be provided. Even if no changes are made, the new encrypted version of this vote will be distinct from the prior version, so no linkage will be possible.

4.2.2 The Interactive Proof Option

Another option would be to allow voters to engage in a more elaborate interactive protocol to demonstrate that the encrypted ballot produced matches the voter's intentions. Unlike the previous option, a more elaborate proof technique can be used which would not invalidate the ballot for casting. The principal drawback of this option is that voters would have to make random selections or pick random values — something which humans are notoriously poor at doing. It is essential to remember, however, that this is not a requirement that would be imposed on all voters but would rather be an option that voters could avail themselves of if they so choose. The timing-based protocol of [BeTu94] will be described here.

The vote creation device creates a set of, perhaps 100, encrypted ballots that all represent votes identical to the original encrypted ballot already in the voter's possession. The device then commits to these additional ballots by printing them

[4]When polling sites cover multiple precincts, it may be necessary for a voter to provide the device with geographic information to select the proper precinct as well as indicating other options such as party affiliation where appropriate. This may be facilitated by voters checking in with poll workers to receive tokens to indicate the ballots that should be used, but there need be no restrictions on eligibility at this point. Anyone should be able to complete any available ballot on any vote creation device at any time during an election.

[5]A cryptographic thumbprint can be produced by applying a one-way hash to the desired data. Its output is usually about 20-30 bytes or approximately 30-50 alphanumeric characters, although thumbprints as short as 15 alphanumeric characters would likely suffice here.

in a location or form that can not be read by the voter.[6] The voter is then given an opportunity to select a random subset of the encrypted ballots.[7] Once the random selection of encrypted ballots is made by the voter, all selected ballots are verifiably decrypted to show that they all indicate the desired vote. Additional data is also released to show that all unselected ballots are encryptions of votes identical to that of the original encrypted ballot — the demonstration of equivalency of the unselected encrypted ballots and the original encrypted ballot in the voter's possession is done in such a way as to not reveal the votes on these ballots. As a final step, the device creates, for each candidate or option not voted for, encrypted ballots and verifiable decryptions thereof that correspond to these other options — one for each ballot that the voter selected to have opened.

The ordering of the above steps is crucial. It is easy for the device to create substitute opened ballots that correspond to a candidate *not* voter for. When these substitute ballots replace the previous opened ballots, the interactive proof appears to be a perfectly legitimate proof that the encrypted ballot originally produced for the voter represents a vote for this alternate candidate. The voter now has two or more proofs of the contents of the encrypted ballot, one for each available candidate. These proofs all appear to be equally valid. However, the only proof that is actually valid is the one in which the additional ballots are committed to prior to the random challenge issued by the voter. The voter knows the order in which these values were produced, but has no way to prove this order to a third party and is thereby protected from coercion. It is therefore essential that the commitments must be recorded on separated slips of paper, cards, or regions of a single readable medium or recorded electronically or magnetically in a manner that does not reveal ordering.

A final option is a variant of the above. If an unpredictable source of random challenge bits can be agreed upon, then voters can be relieved of the burden of having to make random selections and it would no longer be necessary to shield the commitments from the voter prior to the issuance of challenge bits. However, this added physical assumption may not be acceptable.

[6]These additional ballots could be printed behind an opaque screen or on the back of a card or could be written on an electronic or magnetic medium that cannot be read by the voter during this process.

[7]If 100 additional ballots are used, one or more random values totaling 100 bits of data can be supplied to select a subset of the encrypted ballots.

4.2.3 Effectiveness of Auditing

Voters have the opportunity to audit the vote creation process by either creating extra ballots and having them decrypted or engaging in interactive proofs to verify the accuracy of the ballots they actually cast. It seems likely, however, that few voters would avail themselves of these opportunities. This may seem to destroy the reliability of such a system. However, Neff has shown ([Neff03]) that very little auditing is required in order to achieve a high degree of assurance of election accuracy.

In a large election, if a mere thousand random voters, officials, and/or observers avail themselves of the opportunity to each audit a single encrypted ballot, and if all such probes are successful, then there an extremely high probability that the announced tally is within 0.5% of the true tally of voters' intentions. The key factor that makes this work is that the lack of identification of voters by vote creation devices effectively forces the devices to always behave honestly. It is no longer necessary for each voter to independently audit the vote creation devices.

4.3 The View for Voting Officials

The task for election officials is to collect all cast ballots and publish them on a public site. Since they are encrypted, the cast ballots can even carry the names of the voters who cast each one — although the names can be removed if this is preferred. Any interested party may then be offered an opportunity to shuffle the ballot set and provide a proof that the resulting ballot set is equivalent. Each shuffling and its proof of correctness is posted on a public site.

Once the final encrypted ballot set has been produced, any sufficiently large set of election trustees act jointly to decrypt each individual ballot in this set. The decryption process also provides data to allow verification that the decryptions are accurate.

Any officials, observers, and voters who wish to do so can use the public data to verify that each step is correct. Voters can also check that their ballots were included unaltered in the original encrypted ballot set, although they will not be able to identify their own ballots among the decrypted final ballots.

5 Conclusions

The methods described above offer a verifiable election system that is far simpler than other such systems previously suggested. From the perspective of a voter, this system can look almost identical to voting with on a "traditional" electronic voting device. However, added capabilities are included which allow voters who care to do so to check that their votes are properly recorded, cast, and included within the tally. Voters and any other interested parties also gain the capability to check that all votes are associated with legitimate voters and are properly tallied. This end-to-end verifiability is much stronger than traditional election systems in which voters can have confidence that their votes are cast as intended but have no direct means to ensure that their votes are included in the tally or that the tally in any way represents an accurate count of legitimate votes.

References

[Bena87] **Benaloh, J.** "Verifiable Secret-Ballot Elections." *Yale University Ph.D. Thesis YALEU/DCS/TR-561.* New Haven, CT (Dec. 1987).

[BeTu94] **Benaloh, J.** and **Tuinstra, D.** "Receipt-Free Secret-Ballot Elections" *Proceedings of the* 26th *ACM Symposium on Theory of Computing.* Montreal, PQ (May 1994) 544–553.

[BJR01] **Bruck, S.**, **Jefferson, D.**, and **Rivest, R.** "A Modular Voting Architecture ("Frogs")." *Workshop on Theory of Elections.* Tomales Bay, CA (Aug. 2001).

[BoFr97] **Boneh, D.** and **Franklin, M.** "Efficient Generation of Shared RSA Keys." *Proceedings of Crypto '97.* Santa Barbara, CA (Aug. 1997) 425–439.

[CDN01] **Cramer, R.**, **Damgård, I.**, and **Nielsen, J.** "Multiparty Computation from Threshold Homomorphic Encryption." *Proceedings of Eurocrypt 2001.* Innsbruck, Austria (May 2001) 280–300.

[Chau04] **Chaum, D.** "Secret-Ballot Receipts: True Voter-Verifiable Elections." *IEEE Security & Privacy 2* 1, (Feb. 2004), 38–47.

[Chau05] **Chaum, D.** "Recent Results in Electronic Voting." *Frontiers in Electronic Elections.* Milan, Italy (Sep. 2005).

[Chau81] **Chaum, D.** "Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms." *Communications of the ACM 24,* 2, (Feb. 1981), 84–88.

[CGS97] **Cramer, R.**, **Gennaro, R.**, and **Schoenmakers, B.** "A Secure and Optimally Efficient Mult-Authority Election Scheme." *Proceedings of Eurocrypt '97.* Konstanz, Germany (May 1997) 103–118.

[CoFi85] **Cohen (now Benaloh), J.** and **Fischer, M.** "A Robust and Verifiable Cryptographically Secure Election Scheme." *Proceedings of the* 26th *IEEE Symposium on Foundations of Computer Science.* Portland, OR (Oct. 1985), 372–382.

[CRS05] **Chaum, D.**, **Ryan P.**, and **Schneider, S.** "A Practical Voter-Verifiable Election Scheme." *Proceedings of the* 10th *European Symposium on Research in Computer Security.* Milan, Italy (Sep. 2005), 118–139.

[DeFr89] **Desmedt, Y.** and **Frankel, Y.** "Threshold Cryptosystems." *Proceedings of Crypto '89.* Santa Barbara, CA (Aug. 1989) 307–315.

[DiHe76] **Diffie, W.** and **Hellman, M.** "New Directions in Cryptography." *IEEE Transactions on Information Theory.* IT–22, 6 (Nov. 1976), 644–654.

[DLM82] **De Millo, R.**, **Lynch, N.**, and **Merritt, M.** "Cryptographic Protocols." *Proceedings of the* 14th *ACM Symposium on Theory of Computing.* San Francisco, CA (May 1982), 383–400.

[ElGa85] **ElGamal, T.** "A Public-Key Cryptosystem and Signature Scheme Based on Discrete Logarithms." *IEEE Transactions on Information Theory.* IT–31, 4 (Jul. 1985), 469–472.

[FiSh86] **Fiat, A.** and **Shamir, A.** "How To Prove Yourself: Practical Solutions to Identification and Signature Problems." *Proceedings of Crypto '86.* Santa Barbara, CA (Aug. 1986) 186–194.

[Furu04] **Furukawa, J.** "Efficient, Verifiable Shuffle Decryption and Its Requirement of Unlinkability." *Proceedings of PKC 2004.* Singapore (Mar. 2004) 319–332.

[FuSa01] **Furukawa, J.** and **Sako, K.** "An Efficient Scheme for Proving a Shuffle." *Proceedings of Crypto 2001.* Santa Barbara, CA (Aug. 2001) 368–387.

[GJKR96] **Gennaro, R.**, **Jarecki, S.**, **Krawczyk, H.**, and **Rabin, T.** "Robust and Efficient Sharing of RSA Functions." *Proceedings of Crypto '96.* Santa Barbara, CA (Aug. 1996) 157–172.

[GMR85] **Goldwasser, S.**, **Micali, S.**, and **Rackoff, C.** "The Knowledge Complexity of Interactive Proof Systems." *Proceedings of the* 17th *ACM Symposium on Theory of Computing.* Providence, RI (May 1985), 291–304.

[Grot03] **Groth, J.** "A Verifiable Secret Shuffle of Homomorphic Encryptions." *Proceedings of PKC 2003.* Miami, FL (Jan. 2003) 145–160.

[GZBJJ02] **Golle, P.**, **Zhong, S.**, **Boneh, D.**, **Jakobsson, M.**, and **Juels, A.** "Optimistic Mixing for Exit-Polls." *Proceedings of Asiacrypt 2002.* Queenstown, New Zealand (Dec. 2002) 451–465.

[JJR02] **Jakobsson, M.**, **Juels, A.**, and **Rivest, R.** "Making Mix Nets Robust for Electronic Voting by Randomized Partial Checking." *Proceedings of the 2002 USENIX Security Symposium.* San Francisco, CA (Aug. 2002), 339–353.

[Neff01] **Neff, C.A.** "A Verifiable Secret Shuffle and its Application to E-Voting." *Proceedings of the ACM Conference on Computer and Communications Security.* Philadelphia, PA (Nov. 2001), 116–125.

[Neff03] **Neff, C.A.** "Election Confidence: A Comparison of Methodologies and Their Relative Effectiveness at Achieving It." Document available from http://www.votehere.com/.

[Neff04] **Neff, C.A.** "Practical High Certainty Intent Verification for Encrypted Votes." Document available from http://www.votehere.com/.

[PBD05] **Peng, K.**, **Boyd, C.**, and **Dawson, E.** "Simple and Efficient Shuffling with Provable Correctness and ZK Privacy." *Proceedings of Crypto 2005.* Santa Barbara, CA (Aug. 2005) 188–204.

[Pede91] **Pedersen, T.** "A Threshold Cryptosystem Without a Trusted Party." *Proceedings of Eurocrypt '91.* Brighton, UK (Apr. 1991) 522-526.

[PIK93] **Park, C.**, **Itoh, K.**, and **Kurosawa, K.** "Efficient Anonymous Channel and All/Nothing Election Scheme." *Proceedings of Eurocrypt '93.* Lofthus, Norway (May 1993), 248–259.

[RSA78] **Rivest, R.**, **Shamir, A.**, and **Adleman, L.** "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems." *Communications of the ACM 21,* 2 (Feb. 1978), 120–126.

[SaKi95] **Sako, K.** and **Kilian, J.** "Receipt-Free Mix-Type Voting Scheme." *Proceedings of Eurocrypt '95.* St. Malo, France (May 1995) 394–403.

[Sham79] **Shamir, A.** "How to Share a Secret." *Communications of the ACM 22,* 11 (Nov. 1979), 612–613.

[Shou00] **Shoup, V.** "Practical Threshold Signatures." *Proceedings of Eurocrypt 2000.* Bruges, Belgium (May 2000) 207–220.

APPENDIX: A Threshold Encryption System

Many threshold encryption mechanisms are available based on a variety of standard public-key mechanisms (see, for example, [DeFr89], [Pede91], [GJKR96], [BoFr97], [Shou00], and [CDN01]).

While several (including [DeFr89], [GJKR96], and [BoFr97]) are based directly on the well-known RSA cryptosystem ([RSA78]), the Pedersen system ([Pede91]), which is based on the Diffie-Hellman key exchange protocol ([DiHe76]) that pre-dates RSA, is somewhat better suited to these purposes.

The so-called ElGamal encryption scheme ([ElGa85]) is a direct application of Diffie-Hellman key exchange. A large prime number p and a generator g of the multiplicative subgroup of integers modulo p are agreed upon and may be shared by all users of the system.[8] Any user of the system may generate a private secret key s by randomly selecting an integer s such that $0 < s < p$ and publishing the corresponding public key $z = g^s \bmod p$. An integer message M in the range $0 \leq M < p$ can be encrypted by selecting a random integer value r such that $0 < r < p$ and forming the pair $(x, y) = (Mz^r \bmod p, g^r \bmod p)$. With access to the corresponding secret key s, one can decrypt (x, y) by computing $x/y^s \bmod p = Mz^r/g^{rs} \bmod p = Mg^{rs}/g^{rs} \bmod p = M \bmod p = M$. There is no known method for decrypting an ElGamal pair without access to the private secret key s. Indeed, this thirty-year-old assertion is the first and oldest mathematical basis in public-key cryptography.

To improve readability, for the remainder of this section, all of the mathematics will be implicitly performed modulo p unless specifically indicated to the contrary.

An ElGamal encryption pair $(x, y) = (Mz^r, g^r)$ can be randomly re-encrypted by selecting a random integer value r' in the range $0 < r' < p$ and forming the new pair $(x', y') = (xz^{r'}, yg^{r'})$. It can be seen that the decryption of (x', y') produces $x'/(y')^s = xz^{r'}/(yg^{r'})^s = Mz^{r+r'}/g^{(r+r')s} = Mg^{(r+r')s}/g^{(r+r')s} = M$. It is therefore a simple matter for any individual — even one who has no knowledge of the decryption of the ElGamal pair (x, y) to form a new ElGamal pair (x', y') which has the same decryption. Based on the same basic Diffie-Hellman assumption, there is no way for an observer without access to the secret key s to see that (x, y) and (x', y') have the same decryption without assistance. However, this assistance can be provided by simply releasing the value r' which can be used by any observer to check that $(x', y') = (xz^{r'}, yg^{r'})$ and that therefore (x, y) and (x', y') do indeed have identical decryptions.

This random re-encryption can be composed in a variety of ways. For instance, if $(x', y') = (xz^{r'}, yg^{r'})$ and $(x'', y'') = (xz^{r''}, yg^{r''})$ are two re-encryptions of the same ElGamal pair (x, y), then both $(x'y')$ and (x'', y'') will have the same decryptions. They can be shown to both match (x, y) by providing both r' and r'', but this not only shows that (x', y') and (x'', y'') are equivalent to each other but also that they are both equivalent to (x, y). By releasing only the single value $(r' - r'') \bmod (p-1)$. The two ElGamal pairs (x', y') and (x'', y'') can be shown to have the same decryptions without any linkage or association to the original ElGamal pair (x, y).

ElGamal encryption supports random re-encryption nicely, but how is the threshold encryption property achieved?

From the perspective of the party forming an encryption, ElGamal, threshold encryption looks identical to ordinary ElGamal encryption. The only differences are manifested in management and use of the secret decryption key. Suppose, for instance, that two parties shared the decryption function by holding secret values a and b such that $s = a + b \bmod (p-1)$. Then a value $(x, y) = (Mz^r, g^r)$ can be decrypted if the two parties contribute the respective values $A = y^a$ and $B = y^b$. The decryption of (x, y) can then be computed as $x/A/B = Mz^r/y^{ar}/y^{br} = Mg^{rs}/g^{ar+br} = Mg^{rs}/g^{rs} = M$. In a similar fashion, the secret decryption key s can be shared amongst more than two parties by splitting s into as many summands as desired.

This alone, however, is not sufficient to allow decryption by an arbitrary threshold such as any 2 of 3 shareholders. Full threshold encryption is achieved by employing Shamir's threshold scheme ([Sham79]) in which a secret value is shared by using polynomial interpolation and evaluation. A random polynomial $P(x) = a_{k-1}x^{k-1} + a_{k-2}x^{k-2} + \cdots + a_2x^2 + a_1x + s$ is selected where k is the desired threshold number of shareholders required to decrypt, the a_i are randomly selected integers in the range $0 < a_i < p$, and s is the secret value. Parties $\mathcal{P}_1, \mathcal{P}_2, \ldots \mathcal{P}_n$ (with $n \geq k$) each hold a point on this polynomial ($\mathcal{P}_i$ holds the value $P(i)$). It is not difficult to show that any k of these points allow interpolation of the polynomial and discovery of the secret value $s = P(0)$, while even complete co-operation among shareholders with $k - 1$ points give no information whatsoever about the value of s.

As in the prior summation case where the required value g^{rs} can be developed from g^r, $A = g^{ar}$, and $B = g^{br}$ without ever exposing the composite secret s, a formula can be described for com-

[8]A generator is a value g such that $r = p - 1$ is the smallest positive integer such that $g^r \bmod p = 1$.

puting g^{rs} directly from any k of the values $g^{rP(i)}$.

The question remains of how shares of a secret decryption key are to be generated and shared amongst an appropriate set of shareholders. While one could imagine a secure device that performs this task and is perhaps even destroyed afterwards, no such device is necessary. Instead, each of the intended shareholders can generate a secret polynomial of the form described above and create a public commitment to this secret polynomial by revealing g^{c_j} for each coefficient c_j of its secret polynomial. The value of this polynomial at point i is given to shareholder $\mathcal{P}_i$ and can be checked by $\mathcal{P}_i$ by verifying that $g^{P(i)} = \sum (g^{c_j})^{i^j}$. The collection of the polynomials generated by the shareholders is summed to form an aggregate polynomial describing an aggregate secret value s, and the sum of the values received by any shareholder constitute a point on this aggregate polynomial. The public key g^s that is associated with the aggregate secret value s is formed as a by-product of the commitment process as the product of the commitments g^{c_0} of the constant terms of each of the constituent polynomials. This allows the sharing of the private key s and thereby satisfies the final requirement for threshold encryption.