Coercion-free Universally-verifiable Voting over the Web

Vivek Pathak

August 11, 2020

Abstract

We support coercion-free and universally-verfiable voting for lay users over the world-wide-web. Established cryptographic and systems security primitives are adapted and enhanced in order to achieve these objectives. Our voter security model differs from prior research in that it delays and defers the ascertainment of voter legitimacy to a human poll manager. This eliminates the requirement of an identity infrastructure, thereby making it possible for lay users to run cryptographically secure polls on demand.

1 Introduction

The rise of online political discourse has counter-intuitively reduced the ability of people to reason, understand, and arrive at sensible political positions supported by large majorities of people. Even though social networks like Facebook and Twitter have been used for political purposes, their effectiveness has had serious shortcomings. Most often the discussions tend to be shallow, and favor one extreme view or another and thereby fails to represent the broad spectrum of political preferences. This flattening of discourse also has the side effect of silencing voices which may differ from the prevalent majority opinion because of fear of retribution.

Lacking a reliable universally trusted mechanism for calculating the true public opinions is a problem. It creates an incentive for politicians, news media, and even pollsters to use propaganda and fake news in order to *build* public opinions aligned with their own narrow interests.

1.1 Solution approach

Coercion freeness Solving the problem of self-censorship requires the following guarantee of privacy: even though the voters are well known, the choices made by them remain private. Coercion freeness goes further by ensuring that voters can not provide evidence about the choice they made. This protects the privacy of voters by ensuring that an adversary can not deduce the vote count on a subset of voters by colluding with malicious voters. A combination of systems security methods along with a cryptographic voting protocol are used to provide coercion-freeness in our solution [2, 4].

Universal verifiability The problem of potential pollster bias is addressed by having a public bulletin board where all the ballots cast in the poll can be validated. The bulletin board also exposes a computation on the homomorphically encrypted ballots such that the poll result can be be verified by anyone. This universal verifiability of poll results addresses the possibility of the pollster making up poll results other than those derived from the ballots. Non-interactive zero knowledge proofs based on Schnorr protocol [1] are used to support universal verifiability. These proofs are constructed for encrypted ballots as well as for the poll result computation. Having publicly accessible universally verifiable proofs ensures that the disclosed ballots are correctly tallied. The complementary problem of ballot hiding is handled by having the voter and the pollster engage in digitally signed communication during ballot casting. Any divergence from the correct voting protocol is provable and shared with the public.

Deferring voter eligibity decisions Secure voting has traditionally been associated with government or other official elections. Accordingly, prior research and implementations have focussed on enabling official or government elections where establishing voter eligibility is typically the first stage of the process. We believe this choice constrains the existing approaches, thereby limiting their applicability. Our approach is to defer voter eligibility to human judgment. The eligibility decisions are done after the ballots have been cast and the eligibility choices are universally verifiabile.

Enabling large scale secure polling We propose to secure the polling mechanism while leaving the policy on voter eligibilty to human pollsters. Removing the identity insfrastructure as a first class participant makes it easy for unsophisiticated pollsters set up secure polls. Using the previously described building blocks, we propose to allow lay users to setup and run cryptographically secure polls over the world wide web, and yet derive poll results which can pass serious scrutiny. Eliminating the voter pre-verification requirement also makes our solution suitable for *automatic polls* where active human ballot casting can be eliminated entirely. For example, one may automatically calculate the popularity of political positions on online social networks by having a opinion classifier automatically feed its results into a virtual opinion poll. Depending on the application area, an appropriate level of effort can be invested in voter verification in order to operate at a suitable level of accuracy versus convenience trade-off.

2 Preliminaries

Our solution can be described in terms of the participants, the notations used to describe the interactions, and the primitives used to achieve the desired security properties.

2.1 Participants

Pollster service The poll is run by a *Pollster service* application which allows users to interact with it using a RESTful interface. The pollster service implements the following functions:

- Create poll
  Lay users, or *human pollsters*, can define polls. Upon poll creation the pollster service accepts ballots on a voting URL. This voting URL can be shared with prospective voters.

- Accept ballot
  Voters cast ballots using the voting URL RESTful interface exposed by the pollster service. Because the ballot needs to have a well defined structure with cryptographic contents, an accompanying library is supported to make it easy for lay voters to cast their votes.

- Bulletin board
  The pollster service publishes ballots on a bulletin board. Once a sufficient number of ballots have been collected, and any disallowed voters trimmed, a tally of valid votes are also published on the bulletin board.

Table 1: Notation

K_A	Public key of the participant A
K_A^{-1}	Private key of the participant A
$\{x, y, z\}$	A message containing x, y and z
$\{x\}_A$	A message signed by A

Voters The voters are expected to be unsophisticated internet users who cast their ballots through a *votster widget.* The function of this widget is to faithfully convert the voter choice into a anonymity preserving ballot and to post it on the ACCEPT BALLOT interface of the Pollster service.

Verifiers Verifiers may be lay users or sophisticated cryptographers who can rerun the ballots-to-tally computation being exposed on the BULLETIN BOARD interface of the the pollster service.

2.2 Notations

The general notations used in this paper are shown in Table 1. We consider n voters $v_1 \dots v_n$ who want to vote on a binary poll $\mathcal{B}$ with choices 0 or 1. The pollster service is represented as $\mathbf{P}$. It has a well known public key K_P which is used for creating digitally signed messages.

The participants communicate with each other using messages, which are represented using the security protocol notation [1]. For example, A sending a message containing X and Y to B is represented as follows:

$$A \rightarrow B \quad : \quad \{X, Y\}$$

2.3 ElGamal cryptosystem

The ElGamal cryptosystem is a well known and established public key cryptosystem based on the Discrete Logarithms problem [3]. Its security comes from the assumption that computing discrete logarithms over a prime group is hard.

[1] Even though an abstract security protocol notation is used for protocol specification and analysis, the implemented messages may be represented in a more machine friendly manner, for example as JSON objects.

Assumption 1 *Given a large prime p and a generator g, it is computationally infeasible to find x from $y \equiv g^x \bmod p$.*

The ElGamal cryptosystem works by all players initially agreeing to a large prime p and a generator g. The receiver chooses a secret key $s < p$ and publishes the public key $h \equiv g^s \bmod p$, which is assumed to be well-known to all parties.

In the context of the ElGamal cryptosystem we will assume all random choices and operations are done modulo p. Hence one can succintly represent the public key publication message as

$$\text{Receiver} \rightarrow \text{Sender} \quad : \quad \{g^s\}$$

Suppose the sender wants to send a message m to the receiver, then it chooses a random integer r and sends the following message:

$$\text{Sender} \rightarrow \text{Receiver} \quad : \quad \{g^r, h^r m\}$$

Since the receiver does not know about the random choice r made by the sender, we represent the received message as $\{x, y\}$. Using its secret s, the receiver recovers the message by computing $\frac{y}{x^s}$, as shown below.

$$\begin{aligned} \frac{y}{x^s} &= \frac{h^r m}{g^{r^s}} \\ &= \frac{(g^s)^r m}{g^{r^s}} \\ &= m \end{aligned} \qquad (1)$$

2.4 Non-interactive zero knowledge proofs

Zero knowledge proofs permit a prover to convince a verifier about their knowledge of a particular information without divulging that information. The proving process can either take the form of an interactive session where the verifier can issue challenges and the prover can respond correctly. These ineractions can be converted into an equivalent non-interactive proof, thereby avoiding the need for participants to be online. One such construction is the Schnorr protocol, which provides universal verifiability.

The objective of the Schnorr protocol is for the prover to convince a verifier that it knows the discrete logarithm x of a well known value $y \equiv g^x$ *without* disclosing the value x. The interactive protocol proceeds by the prover and verifier selecting random numbers, e and c respectively, and exchanging the following messages:

$$\begin{array}{lcc} \text{Prover} \rightarrow \text{Verifier} & : & \{t \equiv g^e\} \\ \text{Verifier} \rightarrow \text{Prover} & : & \{c\} \\ \text{Prover} \rightarrow \text{Verifier} & : & \{d \equiv e - xc\} \end{array}$$

The verifier checks if $y^c g^d = t$, which follows from :

$$\begin{aligned} y^c g^d &= g^{xc} g^d \\ &= g^{(xc+d)} \\ &= g^{(xc+e-xc)} \\ &= g^e \\ &= t \end{aligned} \tag{2}$$

A non-interactive proof is generated by constraining the verifier challenge c to be predictable by the prover yet to resist an adaptive ciphertext attack. This can be done by having it depend on a secure hash function H whose output is sufficiently random and unpredictable. Then the zero knowledge interactive protocol with the verifier v_i can be transformed into its non-interactive version with a transcript as follows:

$$\{g^e, H(g^e, v_i), e - xH(g^e, v_i)\}$$

Such a transcript can be produced by the prover in advance, without requiring any action from the verifier. This Schnorr protocol construction is used for ensuring universal verifiability of several properties of the voting protocol.

3 Voting Protocol

The voting protocol closely follows the cryptographic setup described in Schoenmaker et. al.'s [2] multi authority election scheme. Given the different threat model, the threshold encryption multi authority features are eliminated and replaced with a single pollster service.

This is justified on the grounds that neither preventing malicious voters from voting nor defending against malicious authorities is a goal of this solution.

The malicious voter handling aspect is left to the human level pollsters and human authorities which can presumably scale according to the level of interest in the specific poll. Our protocol ensures that the set of voting parties, the subset of eligible voters selected after ballot casting, and the decisions about accepted votes are publicly verifiable. Another aspect of differentiation is *coercion-freeness*, which is an important requirement for a trustable voting infrastucture, as described in [4]. This property follows from the fact that only the aggregate totals are ever decrypted, thereby providing voters deniability about their vote, which in turn makes the election coercion-free by allowing the voter to lie about the vote they had cast[2].

Malicious authorities are not considered as a threat because we propose to establish authority honesty via open source implementation as well as massive public validation of the algorithms and artifacts involved. The authority supports the fully open source voting protocol in an open verifiable manner. The voting protocol is explained below in terms of its phases:

3.1 Poll creation

On receiving a request to create a poll, the Pollster service chooses a large prime p, a generator g, and an integer private key $s < p$ large enough such that an $O(\sqrt{s})$ time algorithm is computationally infeasible. The public key $h \equiv g^s \bmod p$ is computed from the private key. The values p, g and h are exposed on the public bulletin board.

This step shall also disclose the non-cryptographic attributes of the poll. For example, the expected number of voters, the duration of the poll, as well as the locations where a vote may be cast and the location of the bulletin board where the results shall be available.

This stage can also be used to cryptographically commit the polling parameters. Accordingly the hash of the contents of the initial bulletin board shall be stored on a blockchain in a future version of our implementation.

[2] Ideally one would add a legal clause at the bottom of the ballot stating that it is your right, and under some circumstances, perhaps your duty, to lie about your vote. Such a disclaimer would make the deniability based coercion freeness obvious to unsophisticated voters. Achieving coercion freeness requires both social and technological actions. Our protocol provides the latter.

Table 2: Values of ballot parameters

Variable	Vote +1	Vote −1
x	g^{α}	g^{α}
y	$h^{\alpha}G$	$\frac{h^{\alpha}}{G}$
a_1	$g^{r_1}x^{d_1}$	g^{ω}
b_1	$h^{r_1}(yG)^{d_1}$	h^{ω}
a_2	g^{ω}	$g^{r_2}x^{d_2}$
b_2	h^{ω}	$h^{r_2}\left(\frac{y}{G}\right)^{d_2}$

3.2 Casting encrypted ballots

The specific values sent out in the encrypted ballots depends on whether the vote is positive or negative. The values of the variables used in casting the universally verifiable ballot are shown in Table 2. The values are expressed in terms of a generator G and random numbers α, r_1 and r_2.

The universally veriable ballot to be cast by the voter v_i can be expressed as a 3-tuple $\{T, C, D\}$, where:

$$\begin{aligned} T &= \{x, y, a_1, b_1, a_2, b_2\} \\ C &= H(T||v_i) \\ D &= \{d_1, d_2, r_1, r_2\} \end{aligned} \tag{3}$$

Any verifier needs to establish that i) the vote itself is valid, i.e. is a +1 or −1, and ii) that the ElGamal pair $(x, y) \equiv (g^{\alpha}, h^{\alpha}m)$ is properly constructed using the publicly known values g and h. These properties follow from testing the following on the encrypted ballot:

$$\begin{aligned} C &= d_1 + d_2 \\ a_1 &= g^{r_1}x^{d_1} \\ b_1 &= h^{r_1}(yG)^{d_1} \\ a_2 &= g^{r_2}x^{d_2} \\ b_2 &= h^{r_2}(\frac{y}{G})^{d_2} \end{aligned} \tag{4}$$

3.3 Counting ballots

Consider the encrypted vote part of the ballot $(x, y) \equiv (g^\alpha, h^\alpha G^m)$ where $m \in +1, -1$. The homomorphic property of the encryption allows us to take the product(modulo the large prime) of the ElGamal pairs, and the resultant values can be decrypted to arrive at the total. Consider several encrypted ballots $(x_i, y_i) \equiv (g^{r_i}, h^{r_i} G^{m_i})$, and let

$$(X, Y) \equiv (\prod x_i, \prod y_i)$$

Then it follows that:

$$\begin{aligned} \frac{Y}{X^s} &= \frac{\prod h^{r_i} G^{m_i}}{\prod (g^{r_i})^s} \\ &= \frac{\prod g^{sr_i} G^{m_i}}{\prod (g^{sr_i})} \\ &= \prod G^{m_i} \\ &= G^{\sum m_i} \end{aligned} \tag{5}$$

Hence the total vote count is given as $\log_G \frac{Y}{X^s}$, which can be calculated by the pollster service as it knows the private key s corresponding to this poll.

References

[1] R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Proceedings of the 14th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '94, pages 174–187, London, UK, UK, 1994. Springer-Verlag.

[2] R. Cramer, R. Gennaro, and B. Schoenmakers. A secure and optimally efficient multi-authority election scheme. In *Proceedings of the 16th Annual International Conference on Theory and Application of Cryptographic Techniques*, EUROCRYPT'97, pages 103–118, Berlin, Heidelberg, 1997. Springer-Verlag.

[3] T. Elgamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, July 1985.

[4] C. Karlof, N. Sastry, and D. Wagner. Cryptographic voting protocols: A systems perspective. In *Proceedings of the 14th Conference on USENIX Security Symposium - Volume 14*, SSYM'05, pages 3–3, Berkeley, CA, USA, 2005. USENIX Association.