Ad Hoc Voting on Mobile Devices

Manu Drijvers, Pedro Luz, Gergely Alpár and Wouter Lueks
Institute for Computing and Information Sciences (iCIS),
Radboud University Nijmegen, The Netherlands.

May 20, 2013

Abstract. We describe an ad hoc voting scheme that allows any group of people to very easily cast electronic votes in ad hoc settings. One of the challenges of online voting schemes is to ensure security and privacy. We propose a new approach, the ad hoc voting pattern, that meets these requirements. To show the feasibility of the proposed solution we implemented such a scheme on the Android platform. Moreover, due to the access point functionality of modern smartphones and the fact that the proposed scheme runs completely on an ad hoc fashion, the implemented scheme does not require internet access.

1 Introduction

The last decade has seen a clear shift from traditional mass communication to modern and personalized interactions. Mobile devices and ad hoc relations play an important role in this shift. As such, new protocols and schemes have been designed in order to provide secure, user-friendly and robust applications. An example of such applications is electronic voting, also known as e-voting, designed to provide an alternative and fast way of voting.

Electronic voting schemes must provide the same security properties as traditional paper ballot voting. The security requirements of such protocols include integrity, voter authentication, and ballot secrecy. These represent a big challenge for ad hoc networks since it is easy to eavesdrop on connections or tamper with protocols by connecting extra devices wirelessly. Integrity is one of the most important requirements for a voting protocol: no one should be able to tamper with votes. Also ballot secrecy is very important, as it enables sincere voting, by mitigating external factors such as coercion, social pressure, intimidation and bribery.

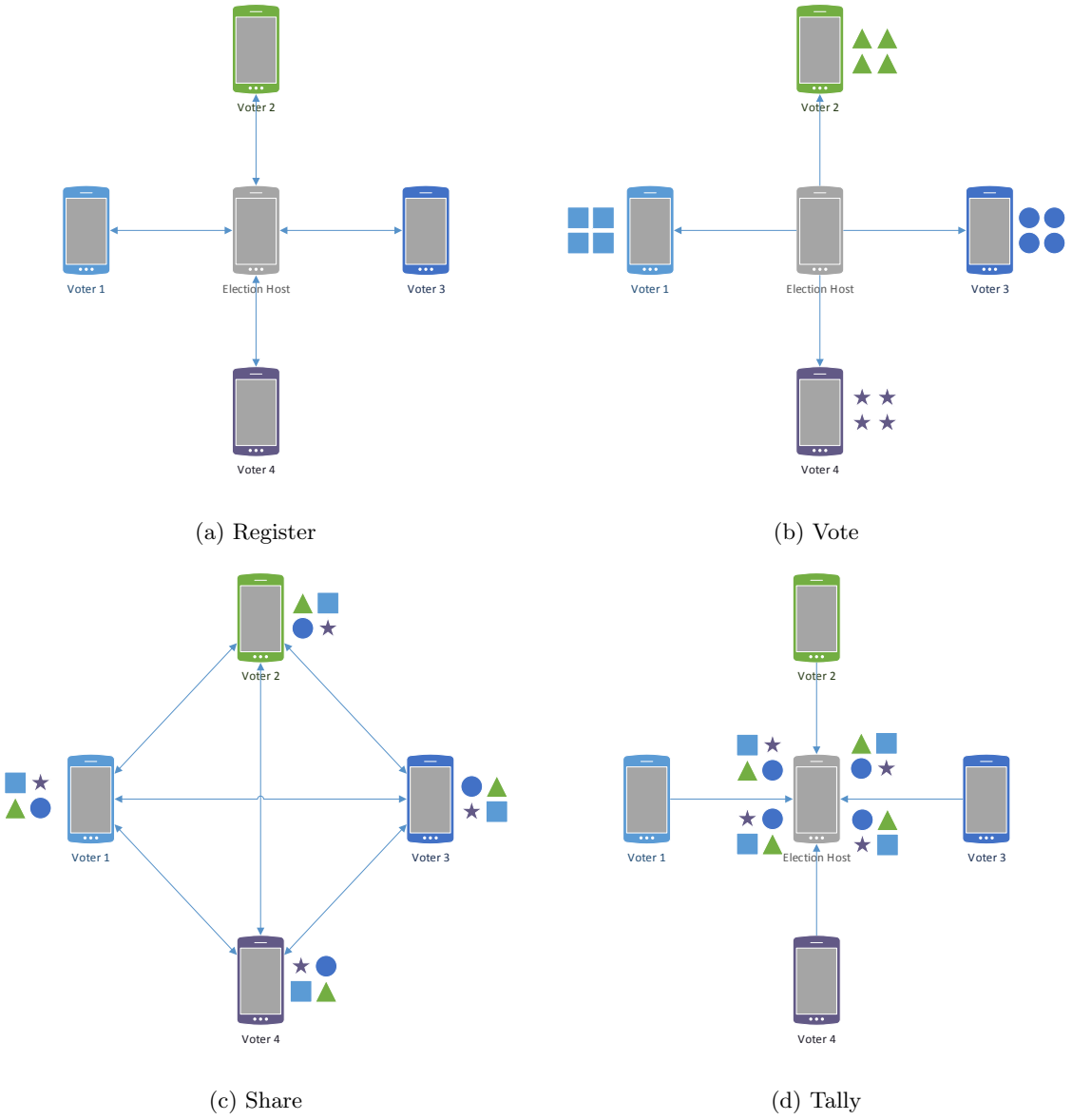

Fig. 1: Ad Hoc Voting Pattern - (a) Registration stage, (b)+(c) Voting stage, (d) Counting stage.

An electronic voting scheme following the ad hoc voting pattern can be thought of as the following three-stage protocol: 1. Acquire an ephemeral identity using a blind signature scheme (Figure 1a); 2. Vote using this identity and distribute the secret ballot using secret sharing (Figures 1b and 1c); and 3. Combine all secret fragments to calculate the result (Figure 1d).

Our work covers the analysis of existing e-voting schemes present in the literature, and a new general ad hoc voting pattern, inspired by Parakh and Kak [11]. We also propose a new scheme, that instantiates this pattern.

Our paper is structured as follows. First, we introduce some of the existing e-voting schemes in Section 2. Next, we generalize Parakh and Kak's [11] proposal to a three-stage pattern, Section 3. We demonstrate the feasibility of the proposed scheme by

an Android implementation described briefly in Section 4. We conclude the paper in Section 5.

1.1 Related Work

Kiayias and Yung [10] define requirements for boardroom elections and a protocol that matches these requirements. This is first improved upon by Groth [8] and then by Hao et al [9]. However, even their scheme [9] becomes inefficient for elections with many candidates, as the computational complexity of tallying grows exponentially in the number of candidates.

Helios [1] is a web-based implementation of an e-voting scheme based on Benaloh's simple verifiable elections [4]. This implementation has shown to be successful [2]. A downside of Helios is that it needs an external voting server to host the election.

All of the voting schemes mentioned above use some form of ElGamal encryption [7], due to its homomorphic properties, and zero-knowledge proofs [6]. The latter are typically used to show that ballots were formed correctly.

Parakh and Kak proposed a voting scheme based on implicit data security [11]. The scheme has security flaws of which the most devastating is that anyone can vote multiple times because of a broken signature scheme. Despite its weaknesses, their work inspired the ad hoc voting pattern proposed in this paper.

2 Technical Background

There are several cryptographic primitives that allow one to create secure e-voting schemes. In this section we provide a description of two different primitives: Shamir's secret sharing scheme and Schnorr's blind signature scheme.

2.1 Notation

In the remainder of this paper we will use the following notation. We write $a \in_R A$ to denote that a is chosen uniformly at random from the set A. Furthermore, we write $\mathbb{Z}_q$ for the group consisting of the integers modulo a prime q.

2.2 Shamir's Secret Sharing Scheme

Shamir's secret sharing scheme can be used to make t-out-of-n sharing of a secret s. Such a sharing consists of n shares, one for each party. The secret s can only be recovered when at least t parties combine their shares. The mathematical principle behind Shamir's secret sharing is that to reconstruct a polynomial of degree $t-1$, one needs at least t points on this polynomial. By encoding the secret in the polynomial, usually as the constant coefficient, one obtains a secret sharing scheme.

2.2.1 Distribution More precisely, the Shamir's Secret Sharing Distribution algorithm $SSSD(s,t,n)$ to create a t-out-of-n secret sharing of a secret $s \in \mathbb{Z}_q$ consists of the following steps.

1. Choose coefficients $a_1, \ldots, a_{t-1} \in_R \mathbb{Z}_q$ and define the degree $t-1$ polynomial by $f(x) = s + a_1 x + \ldots a_{t-1} x^{t-1}$.
2. Create the n secret shares s_i by setting $s_i = f(i)$.

2.2.2 Reconstruction Given a subset of t shares s_i, $i \in \mathcal{I}$ it is possible to reconstruct the polynomial f and hence recover the secret s. Generally, the polynomial is reconstructed using Lagrange polynomials, but since we are only interested in $f(0)$ we only need the following Lagrange coefficients:

$$l_j^{\mathcal{I}} = \prod_{i \in \mathcal{I} \setminus \{j\}} \frac{i}{i-j}$$

Then the secret s can be recovered by setting $s = f(0) = \sum_{i \in \mathcal{I}} l_j^{\mathcal{I}} s_i$.

2.3 Schnorr Blind Signature Scheme

In traditional digital signatures the signer knows the message it is going to sign. However, in a blind signature scheme the signer can sign a message without knowing that message. We use this in our voting pattern because this allows us to decouple registration, where you are identifiable, from voting, where you want to be anonymous.

The first example of a blind signature scheme is described by Chaum [5]. In this paper we use Schnorr's blind signature scheme. It is a variant of Schnorr's identification protocol. Consider a signer with private key x and public key $h = g^x$. In Figure 2 we show how this signer blindly signs a message m held by the recipient. In the following we write $(c', r') = SBSSign(x, m)$ to denote this interactive protocol. The blind signature can then be verified by checking whether $c' \stackrel{?}{=} \mathcal{H}(g^{r'} y^{-c'} \bmod p \| m)$. This scheme is provably secure if $\mathcal{H}$ is modelled as a random oracle [3].

Signer Secret: x		**Recipient** Message: m
$w \in_R \mathbb{Z}_q^*$		
$a := g^w \pmod p$	$\xrightarrow{a}$	$\alpha, \beta \in_R \mathbb{Z}_q$
		$a' := a \cdot g^\alpha y^\beta \pmod p$
		$c' := \mathcal{H}(a' \| m)$
	$\xleftarrow{c}$	$c := c' + \beta \pmod q$
$r := c \cdot x + w \pmod q$	$\xrightarrow{r}$	$a \stackrel{?}{\equiv} g^r \cdot y^{-c} \pmod p$
		$r' := r + \alpha \pmod q$
		Signature: (c', r')

Fig. 2: Messages exchanged between *Signer*, with private key x and public key $y = g^x$ and *Recipient* in order create a blind signature of the message m. The values p, q, g are system parameters[1], and $\mathcal{H}$ is a cryptographic hash function mapping strings onto $\mathbb{Z}_q$.

2.4 Voting scheme properties

Despite the variety of e-voting schemes that exists, most try to achieve the same set of security properties. The security properties for electronic vote can be categorized in two groups: correctness requirements and privacy requirements. A reliable scheme must provide the same level of correctness as the traditional paper ballot voting. The idea of reliable systems is based on the accuracy of the scheme, its integrity and democracy.

[1] An instantiation of such parameters can be found in Section 3.2.1.

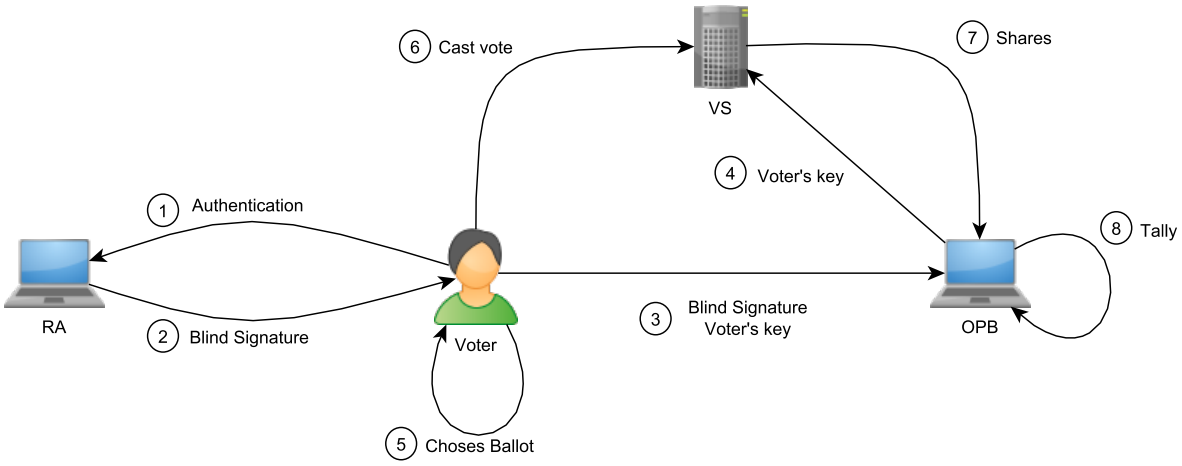

Fig. 3: The Ad Hoc Voting Pattern in 8 steps. The registration authority (RA) registers eligible voters, the online polling booth (OPB) enables voting and tallies the votes afterwards, and the voting servers (VS) register the votes.

On the other hand, a system can be considered as privacy-friendly if it assures ballot secrecy and anonymity.

Schneier [12], introduced these security properties in the scope of the electronic voting by defining the following requirements: ***S1*** - Only authorized voters can vote and at most once (eligibility)[2]; ***S2*** - No one can determine what anyone else voted (ballot secrecy); ***S3*** - No one can change anyone else's vote without being discovered (integrity); ***S4*** - No one can duplicate anyone else's vote (integrity); ***S5*** - Each voter can verify that his/her vote was counted (verifiability).

3 Ad Hoc Voting scheme

The proposed solution is a voting scheme that operates in an ad hoc scenario. In this section, we start by providing an overview of the ad hoc voting pattern, followed by an instantiation. Finally, we analyze the security of the scheme.

The system contains four types of parties: voters, one registration authority (RA), one online polling booth (OPB) and n voting servers (VS). In the following we assume that we always require at least t voting servers to provide final results. For an ad hoc voting scheme, we propose to have every voter to simultaneously act as a voting server.

3.1 Ad hoc voting pattern

The proposed voting scheme can be described as a three stage protocol. The steps of each stage are depicted in Figure 3.

The first stage is the registration stage. First the voter authenticates him- or herself to the *registration authority* (step 1). If the voter is eligible, the RA will blindly sign a random identity generated by the voter (step 2).

[2] Schneier divides ***S1*** into two different requirements. We decided to formalize it as one since both requirements are related to eligibility.

The second stage is the voting stage. The *online polling booth* (OPB) will verify the blind signature to determine eligibility (step 3). The blindness of the signature ensures that the OPB and the RA cannot collude to recover the identity of the voter. Only the random identity is known to the OPB. Also, the voter will send a key to the OPB (step 3), which will then be forwarded to the *voting server*s (VS) (step 4), to allow the voter to access the VSs. Finally, the voter creates its ballot and creates n shares using a t-out-of-n secret sharing scheme (step 5). It uses its key to send these shares securely to each of the VSs (step 6).

When all votes have been cast the OPB closes, and initiates the counting phase. The OPB collects at least t shares of each ballot and reconstructs the original vote (step 7). Each ballot is counted and the final result is published (step 8). In the end, the secret shares are published allowing the voters to verify that their ballot was correctly counted.

3.2 Ad hoc voting scheme

After a high-level overview over the ad hoc voting pattern, we provide the detailed description of scheme, where we use explicit cryptographic primitives. We start by setting up the system parameters necessary for the cryptographic primitives and then we proceed with the description of each stage.

3.2.1 Set up To setup the system, first generate two primes p and q of 1024 and 160 bits respectively, such that q divides $p-1$. Next, fix a generator $g \mod p$, such that g has order q in $\mathbb{Z}_p$, and publish the parameters p, q and g. Next, the RA generates a private key $x \in_R \mathbb{Z}_q$ and publishes its public key $y = g^x \mod p$. Furthermore, each of the voting servers generates a private key u_i and publishes its public key $h_i = g^{u_i} \mod p$.

3.2.2 Registration stage In the registration stage the voter authenticates itself with the RA and obtains a blind signature on a random identity that will be used to cast the vote. Concretely this can be implemented as follows.

1. The voter authenticates itself to the RA and sends it a random identity $r_{id} = g^r$ with $r \in_R \mathbb{Z}_q$.
2. The RA verifies the eligibility of the voter, and aborts if the voter is not eligible.
3. The voter and the RA simultaneously execute the $SBSSign(x, r_{id})$ protocol. In the end the voter has a signature (c, r) on its random identity.

3.2.3 Voting stage At the beginning of the voting stage the voter creates an ephemeral private key v_i and sets its public key to $V_i = g^{v_i} \mod p$. It sends its random identity r_{id}, the signature (c, r) on that identity, and its public key to the OPB. If the signature is valid, the OBP forwards the public key to the VSs to allow the identity r_{id} to cast a vote. The VSs only accept these messages from the OBP.

To cast its ballot b, the voter creates Shamir secret shares $(b_1, \ldots, b_n) = SSSD(b, t, n)$ of this ballot. Then, it calculates a shared key $k_i = h_i^{v_i}$ with voting server i. First, the voter identifies itself to voting server i by sending r_{id}, and then it sends the share b_i encrypted using k_i. The voting server can derive the same key by setting $k_i = V_i^{u_i}$, and thus can store the vote (b_i, r_{id}).

3.2.4 Counting stage In this phase, the OPB collects all the ballot pieces from t VSs and reconstructs the ballots using Shamir's secret sharing reconstruction scheme. Now, the OPB simply tallies the ballots and publishes the result together with the polynomials (indexed by r_{id}). This allows every user to verify that his vote was correctly counted.

3.3 Analysis of security requirements

In section 2.4 we presented a list of requirements for electronic voting schemes. We now show how our ad hoc voting scheme meets such requirements.

The requirement ***S1*** is met because to cast a vote one needs a signed anonymous identity, and only the RA can produce these signatures. The RA will only sign an anonymous identity for eligible voters. Furthermore, the RA will only sign one anonymous identity for each voter, and with a single anonymous identity only one vote can be cast.

A voter only shows its true identity to the RA, over a secure channel. This means that the RA is the only one that knows a true identity. However, the RA never learns the anonymous identity of the voter, because it blindly signs it. All other steps always refer to this anonymous identity, hence the true identity remains hidden. Thus requirement ***S2*** is satisfied.

At least t VSs need to cooperate in order to change a ballot. The properties of the secret sharing scheme ensure that any smaller coalition can only randomize the ballot, which with only very low probability will give a valid ballot again. So requirement ***S3*** is also satisfied.

As long as less than t VSs are corrupted, duplicating someone else's vote is impossible, ***S4***. To duplicate a vote, one has to know the vote. This vote is distributed over all VSs, and t shares are required to reconstruct the vote. If less than t VSs are corrupted, the vote cannot be determined and can therefore not be copied.

At the end of the voting phase, the OPB publishes the final result, as well as a list of all the polynomials received. This allows every voter to verify whether his vote, identified by the polynomial, is included in the list (***S5***). Furthermore, each voter can verify if all the reconstructed polynomials contain a valid $f(0)$. This allows a voter to verify that the sharing and reconstruction algorithms for all polynomials were correctly executed and that no one tried to tamper individual shares, ***S3***.

4 Implementation

We implemented the ad hoc voting scheme on the Android platform. One device will act as the host, and perform the tasks of the RA and OPB. Having one device carrying out both tasks is secure under the assumption that the communication channel does not reveal anything about the identity. This device is excluded from voting. Every voter will also act as VS. TCP connections are used for all the communication. In practice one should use transport layer security, which has been omitted for this proof-of-concept. The threshold used in Shamir's Secret Sharing is equal to the number of voting servers, which means that, unless all voting servers are corrupted, ballots cannot be reconstructed. Our proof-of-concept uses open authentication, which means that anyone able to connect to the RA is able to vote. Since modern smartphones can

function as a Wi-Fi access point, an ad hoc network hosted by one of the smartphones can be used to perform the voting.

5 Conclusion

We have proposed the ad hoc voting pattern, a new approach to ad hoc voting based on anonymous identities and secret sharing. An instantiation of this pattern has been given, based on Schnorr's blind signatures and Shamir's secret sharing, which fulfils desired properties such as ballot secrecy, eligibility, integrity and verifiability. One can easily think of other instantiations using different cryptographic primitives. For future work it would be interesting to analyze different instantiations and analyze their security. This might allow us to make the security requirements posed on the blind signature scheme and the secret sharing scheme more explicit. One could also research different authentication models, for example using out-of-band channels like NFC. Finally, it would be interesting to make a fully secure implementation of the ad hoc voting scheme.

References

1. Ben Adida. Helios: web-based open-audit voting. In *Proceedings of the 17th conference on Security symposium*, SS'08, pages 335–348, Berkeley, CA, USA, 2008. USENIX Association.
2. Ben Adida, Olivier De Marneffe, Olivier Pereira, and Jean-Jacques Quisquater. Electing a university president using open-audit voting: analysis of real-world use of helios. In *Proceedings of the 2009 conference on Electronic voting technology/workshop on trustworthy elections*, pages 10–10, 2009.
3. Mihir Bellare and Phillip Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *Proceedings of the 1st ACM conference on Computer and communications security*, CCS '93, pages 62–73, New York, NY, USA, 1993. ACM.
4. Josh Benaloh. Simple verifiable elections. In *Proceedings of the USENIX/Accurate Electronic Voting Technology Workshop 2006 on Electronic Voting Technology Workshop*, pages 5–5, 2006.
5. David Chaum. Blind signatures for untraceable payments. In *Advances in Cryptology: Proceedings of CRYPTO '82*, pages 199–203. Plenum, 1982.
6. Ronald Cramer, Ivan Damgrd, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In YvoG. Desmedt, editor, *Advances in Cryptology CRYPTO 94*, volume 839 of *Lecture Notes in Computer Science*, pages 174–187. Springer Berlin Heidelberg, 1994.
7. T. Elgamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *Information Theory, IEEE Transactions on*, 31(4):469 – 472, jul 1985.
8. Jens Groth. Efficient maximal privacy in boardroom voting and anonymous broadcast. In Ari Juels, editor, *Financial Cryptography*, volume 3110 of *Lecture Notes in Computer Science*, pages 90–104. Springer Berlin Heidelberg, 2004.
9. Fao Hao, Peter YA Ryan, and P Zieliński. Anonymous voting by two-round public discussion. *IET Information Security*, 4(2):62–67, 2010.
10. Aggelos Kiayias and Moti Yung. Self-tallying elections and perfect ballot secrecy. In David Naccache and Pascal Paillier, editors, *Public Key Cryptography*, volume 2274 of *Lecture Notes in Computer Science*, pages 141–158. Springer Berlin Heidelberg, 2002.
11. A. Parakh and S. Kak. Internet voting protocol based on implicit data security. In *Computer Communications and Networks, 2008. ICCCN '08. Proceedings of 17th International Conference on*, pages 1 –4, aug. 2008.
12. Bruce Schneier. *Applied Cryptography: Protocols, Algorithms, and Source Code in C*. John Wiley & Sons, Inc., New York, NY, USA, 2nd edition, 1995.