Secure Voting From Your Living Room

David Andreu, Alex Escala, Guillem Calduch

E-mail:

ABSTRACT

Nowadays, e-voting is used in more and more countries and in all kind of elections. There's not a single solution to e-voting and each has different characteristics and areas of application. In this article we give a brief description of what we understand by e-voting systems and we introduce the different models used in remote electronic voting. We explain how these systems work and give some ideas of their advantages and drawbacks. Finally we give some examples of elections using e-voting and some ideas of the challenges that researchers try to overcome.

1. WHAT IS E-VOTING?

There are several definitions about what we call "e-voting". In a broad sense, "e-voting" is considered the introduction of electronic systems in an electoral process, without discerning if they take part in the electoral roll, in making the district maps, in the electoral logistics, in the mechanisms of voting or in the count and transmission of the results. However, in this article we will only focus on two areas of application: the emission of the votes and the subsequent count of them.

There is not a unique way to implement an e-voting system. In fact, there are three types of implementations, which basically differ in their benefits and risks. These types are listed below:

A) Optical Mark Recognition (OMR): systems of automatic count of votes by using techniques of optical recognition, which can read marks in the ballots made by the voters. These systems are focused on the count.

B) Direct-Recording Electronic (DRE) Voting System: systems that use digital mechanisms for selecting the vote, such as buttons or a tactile screen. Normally it prevents errors made by the voters because it guides the user step by step. Commonly the votes are registered by the machine but there are some versions that also print a ballot to be placed in a ballot box, usually to check correctness of the election.

Figure 1. A DRE machine.

C) Remote Electronic Voting System: there are some different channels to transmit a vote in a remote way such as Internet (web or e-mail), SMS and others. This kind of systems is more complex than the in-person ones because the electoral authority cannot control all the steps as before.

2. PROS AND CONS OF E-VOTING

It's clear that e-voting can be an improvement on traditional elections for its different features: speed, accessibility, error prevention (which implies a reduction of invalid votes), cost reduction and the possibility that voters verify the correct treatment of their votes.

Even more, e-voting can reduce the cost of large scale elections: there will be one paper ballot per voter (or even no paper ballots at all), in contrast to traditional elections where lots of paper ballots are printed, more than the population who can vote.

In spite of the improvements that e-voting presents, there are some vulnerabilities of e-voting systems that have stopped the extensive use of them. For instance, the virtual nature of the ballots makes that they can be added, manipulated or deleted. Another factor is that the systems used may have problems and errors, compromising voter privacy.

To mitigate these risks cryptographic protocols need to be used along with software auditions. With the second ones we can ensure that the system works as intended, with the first ones we can ensure the following requirements: vote integrity, authentication and privacy of voters, accuracy of

election results and prevention of coercion and vote-selling. Even more, we can also make that individual voters can verify their own vote and the correctness of the result.

3. ELECTRONIC VOTING PROTOCOLS

Now we will present the electronic voting protocols that use advanced cryptographic techniques to fulfill the requirements listed above. These protocols can be classified by the stage in which the voter anonymity is protected:

• Before the voting process:
- *Pre-encrypted ballots* (also known as pollsterless): the voting options (e.g. candidates) are pre-encrypted generating unique individual ballots for each voter.

• During the voting process:
- *Two-agencies model:* it uses a technique called blind signature in order to split the authentication of the vote from the casting of the (validated) vote.

• After the voting process:
- *Mixing model:* votes are shuffled secretly and, at the end of the election, decrypted, to break any correlation between the encrypted votes and the decrypted ones.
- *Homomorphic model*: its objective is to obtain the tally of the elections without decrypting any single vote.

4. PRE-ENCRYPTED BALLOTS

A problem that can arise when using remote electronic voting is that the software used for voting turns malicious. In this way, an intruder could see someone's vote and even send a different vote to the election server, thus breaking privacy and vote integrity.

In 2002, California Internet Voting Task Force and Oppliger presented a solution to that problem: start making use of code sheets (currently known as pre-encrypted ballots). These code sheets contain a voter identifier and codes related to the voting options (each one of these options has a code associated) and, basically, the idea is to deliver a unique code sheet to every voter. The voter will make his selection by introducing the code associated to the alternative he wants to vote for. This way, as nobody except the voter knows the relation between codes and candidates, malicious software won't be able to manipulate the vote.

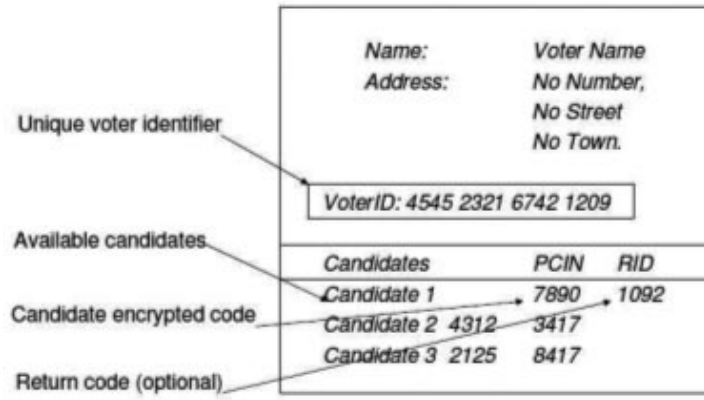

Figure 2. One example of a code sheet.

One consequence of not having to compute any encryption is that votes may be cast from devices with low computing power, such as sending an SMS from a mobile phone.

As we can see in the picture above, there are some return codes: these are to verify that the voting server has received his vote properly. This helps to counter software malfunction or attacks on the voter. One example of an attack is the following: when the voter casts the first vote and every time he casts the same vote the attacker denies the communication between the voter and the voting server. The voter may then send another code (to check if the system works) and the attacker would permit the communication, so finally the voter isn't voting what he wanted to vote. Using return codes this attack is countered because the voter will know if he is receiving a denial of service attack.

All the codes are generated using cryptographic and coding tools and a different ballot will then be secretly sent to each voter. So this is the main drawback of the system: we need to deliver the vote in a secure way and there can be logistics problems when distributing the ballots.

5. TWO AGENCIES MODEL

One of the requirements of e-voting is voters' privacy, but we need to authenticate them. One way to make both things compatible is to split voters' authentication from the ballot casting process to prevent the association between the voter and the vote. To achieve this, the scheme needs to use two different systems.

On one hand, the validation server authenticates the voters and validates the votes in an anonymous way. On the other, the voting server receives the validated votes without identifying the voter and stores the votes. Normally, this kind of

schemes are based on protocols that use a mechanism called blind signature in order to validate votes without compromising the voter privacy.

The first blind signature protocol was designed by David Chaum in 1982. It allows someone to obtain a message signed by another entity without revealing the message contents to the entity. We require that this signature is able to be verified by others (so it's a signature), like the regular digital signatures, and we also require that whoever has signed the message must not be able to guess the message contents (so we call it blind).

The voting system works as follows: the voter encrypts his vote with the public key of the voting server, blinds the message and sends it to the validation server. The validation server will sign the message and the voter will be able to unblind the message and send it to the voting server, who will check the signature, store the vote and finally decrypt all the votes. To clarify it, we present a diagram of the scheme.

In the figure, $E_s(m)$, $D_p(m)$ and $S_s(m)$ stand for encryption using the secret key s, decryption using the public key p and signature using the secret key s, respectively. (Recall that, in asymmetric encryption, a pair of keys {public key, secret key} are used; the first one used to encrypt and the second one to decrypt or make a signature on a message. Decryption and signing are two processes which are usually the same mathematic function).

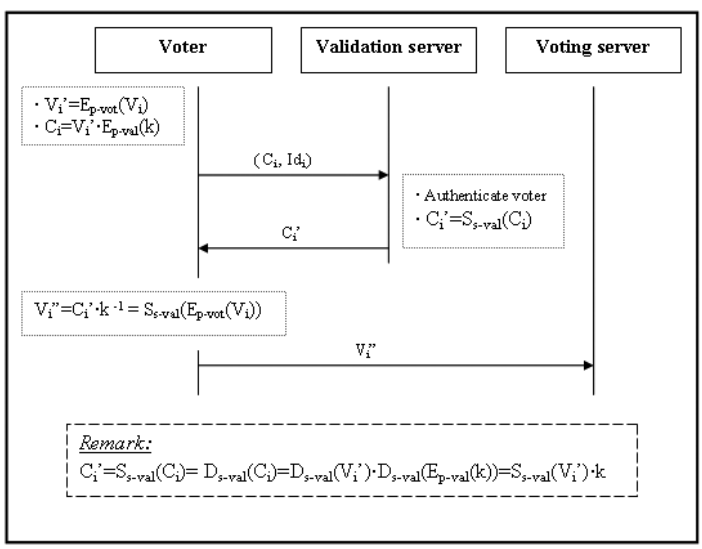

Figure 3. A two-agencies protocol scheme.

Some issues have to be addressed, though. This system breaks the correlation between signature and voting process, so we need to ensure that nobody can monitor both channels and associate the voter with its vote. Even more, the two trusted authorities may be corrupt, so we must use techniques of secret sharing (for the decryption key) and multiparty computation (for the signature). Finally, nobody should be able to vote twice with the same signed message, we can use some cryptographic techniques to avoid this too.

6. MIXING MODEL

Taking another approach to ensure voter's privacy, we could think what happens in traditional elections: when a vote is casted it goes into an urn and when votes are counted there's no way to know the relation between a vote and the voter who cast it because shuffling the urn breaks any correlation between the vote and the voter. This model tries to emulate this behavior.

Actually, this model doesn't come from e-voting. It was conceived to guarantee user anonymity when sending off emails but it has been proved that it can also be applied to electronic voting for the same purpose, guarantee voters' anonymity. In this system we create an anonymous communication channel that will break the correlation between the incoming and outcoming messages.

In mix-nets (a particular case of mixing model) the channel consists of some servers, which are known as mix servers. The first of these servers receives several messages from different senders and applies some sort of transformation on them. Afterwards transformed messages are shuffled randomly and sent to the next server. This one and every remaining server will repeat the same process, delivering messages once more transformed and shuffled to the server that follows them. The reason for not using just one server is that if the server cheats then anonymity is broken.

The messages' transformation can be of two types, which define two types of mixing: decryption mixing and re-encryption mixing. In the first one each mix server decrypts the message that it receives with its private key, so the user has to encrypt the message as many times as the number of servers in the mix-net and a pair of keys must be created for each server. In re-encryption mixing the server just encrypts the message once and each server re-encrypts the message, all encryptions using the key of the recipient. The receiver will then decrypt the message once if a homomorphic cipher is used (which we will explain later).

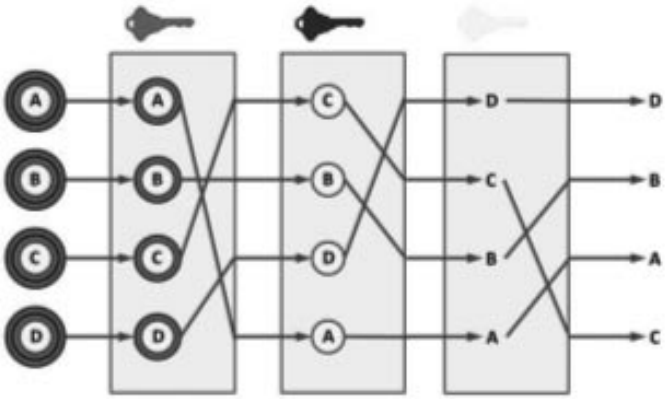

Figure 4. A decryption mix-net.

The main problem of this model is that verifiability by individual voters has to use techniques such as proofs of knowledge (this is, an interactive protocol where the prover convinces a verifier that he knows some values without revealing them), which are computationally expensive.

7. HOMOMORPHIC MODEL

The last model we will talk about involves more cryptographic tools than the other does, which makes it an expensive model in terms of computational power. Using this scheme we can obtain a high level of security and a good verifiability by individual voters. As said in the introduction, in this model only the tally is decrypted, while individual votes are kept encrypted.

Before explaining this model we should introduce two concepts: homomorphic ciphers and secret sharing schemes.

We speak of *homomorphic ciphers* when one can perform some specific algebraic operation on the plaintext by performing a (possibly different) algebraic operation on the ciphertext. For example: multiplying two ciphertext will give us the encryption of the sum of the plaintexts. While this characteristic isn't desirable in data transmission because homomorphic encryptions are malleable, it's very interesting in e-voting schemes. An example of homomorphic cipher is ElGamal.

Homomorphic ciphers are used in the following way: each voter encrypts its vote and publishes it. When all votes are published everybody can compute the encryption of the tally by multiplying the ciphertexts (which will bring to the sum of the plaintexts, this is, the votes). Later we will show how to decrypt the tally.

Another concept we should introduce is secret sharing schemes. The main idea is divide a secret s into n pieces, called shares. We distribute the shares among n users and the scheme is made so that a subset can recover the secret if its members collaborate but a non-authorized subset won't learn anything about the secret. One of the first secret sharing schemes was introduced by Shamir in 1979 and it's a threshold secret sharing scheme. That is, a set is able to reconstruct the key if and only if it consists of more than t people, where t is the threshold specified.

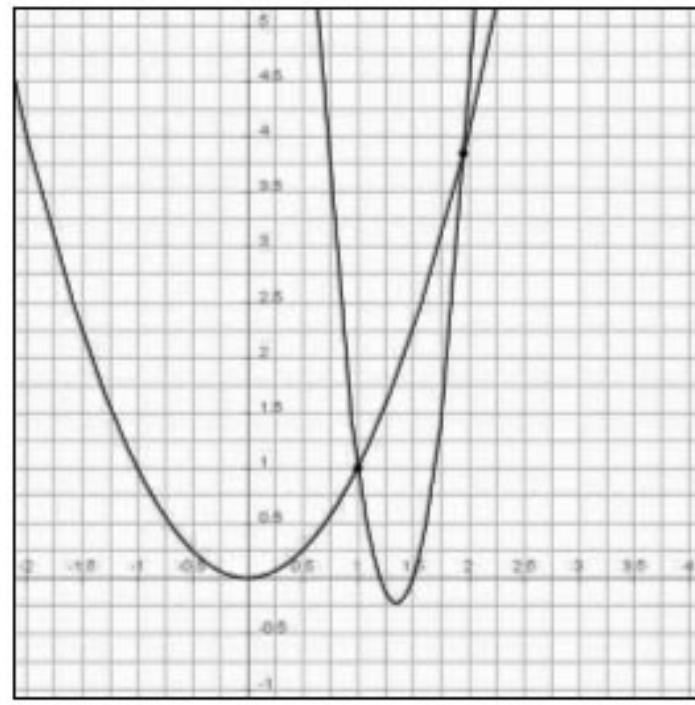

Figure 5. 2 points don't determine a parabola.

The main idea behind Shamir's secret sharing scheme is polynomial interpolation. We know that a polynomial of degree t is determined by t +1 points, so we can give an evaluation of the polynomial to each participant in the secret sharing. When more than t people want to get the secret they can interpolate the polynomial which hides the secret but t people or less won't have any idea of what polynomial is used. This is useful when you need to trust some people and you consider that it's improbable that a large group of people will collude and cheat.

Now we can sketch how the system works. First of all the keys and the shared secret are generated, which can be done by a trusted third party (TTP) or by all the authorities jointly (using multiparty computation). The voters will then encrypt their vote and publish it. When all votes are cast, each authority will compute the encryption of the tally (multiplying the encrypted votes) and will publish their partial decryption with their share of the secret. After that everybody can compute the decrypted tally with the partial decryptions.

Due to the homomorphic properties in this system, we have to check that nobody cheats. For example, a voter could give 100 votes to a candidate (which would add 100 votes to the tally) so we have to avoid this using proofs of knowledge. The same goes for the partial decryptions: we have to make sure that they are done correctly (again, with proofs of knowledge). This means that the computational power needed in this system is higher than in other cases, even for the voters.

8. REAL EXPERIENCES AROUND THE WORLD

There have been electronic elections in lots of countries. Most times these were just non-binding tests, because there was no legislation going over it or just to check if the system worked properly.

In the election to the parliament of Catalonia there was a non-binding pilot test, the first in Spain when speaking of public elections. This was made to test advantages, usability and reliability of this technology. It has also been used in non-governmental elections in Catalonia. One example was the election of the labor union of the autonomic police, where e-voting was used successfully.

We can't speak of more experiences in our country because e-voting is still in a very initial state. However in other places there are lots of experiences and problems related to it.

In Brazil, for example, all votes are cast by electronic voting machines. Use of electronic voting technology was authorized in 1996 in the municipal elections. In 1998 this use was extended when 57% of the electorate used electronic voting. By 2000 Brazilian government had converted to fully electronic voting and deployed over 400.000 kiosk-style machines in elections that year.

An interesting example of how things, if they aren't done properly, won't work is Netherlands. Voting machines were introduced there in 1965, with a good acceptance. In 2006 voters from abroad used Internet to vote, and there were plans to use internet for voters within country. But legislation wasn't as hard as it should have been and machines weren't updated with regard to security provisions. An organization against e-voting appealed that the system was easy to hack and six weeks before parliamentary elections in November 2006 they proved that they could manipulate the machines and they could read them from distance (thanks to Tempest). Needless to say, Internet voting was discarded and classic voting systems were recovered.

But with no doubt, the place where there have been more experiences (and also more problems) is America. On November 4, 2003 in Fairfax County, Virginia machines malfunctioned and collapsed the modems. When 953 voting machines tried to call simultaneously to inform of the results they caused a denial of service accident in the election. Another example: on August 1, 2001 in the Brennan Center at New York University Law School. NY University Law School released a report with more than 60 examples of e-voting machine failures in 26 states in 2004 and 2006. Examples included Spanish language ballots that were cast by voters but not counted in Sacramento in 2004.

In fact, this kind of problems discourages lots of voters, who won't have confidence on this system for many years. For that reason it is important that, when implementing e-voting systems, this is done slowly and carefully to ensure the security of e-voting. There's a need to legislate e-voting with all the details to prevent fraud and malfunctioning of e-voting systems.

9. CURRENT RESEARCH ON E-VOTING

E-voting isn't a closed topic: there're still several factors that can be improved.

One of the research objectives is to reduce computational cost on e-voting solutions. This is very important mostly on the homomorphic model as the cost of its protocols is high. For example, there's research on validity check, the main efficiency bottleneck that limits the application of this kind of e-voting. There are proposed solutions making validity highly efficiency.

Another example is research done in mixing models or shuffling models. There're some papers proposing new shuffling systems trying to speed up the whole process while still having universal verifiability. Another goal is to make proofs of knowledge to the mixing servers in such way that the computational cost is reduced. As an example, B. Shoenmakers and others proposed a protocol using the DFT to have efficient proofs of knowledge.

Apart from research done by universities there also exist some companies that research and provide solutions to be applied in e-voting. One example in Catalonia is Scytl, created in 2001 as a spin-off from UAB students. It offers solutions for e-voting with several applications that go from

government elections to college elections. Scytl works for the Generalitat de Catalunya, the Ministry of Justice of UK and many others. It also collaborates with universities from Catalonia in their research such as UAB or UPC.

AUTHORS

Alex Escala was born in Barcelona, on 1986. He is studying the last course of mathematics degree and telecommunication engineering in the Universitat Politecnica de Catalunya. He is also working in a Department of Applied Mathematics, researching on cryptography.

David Andreu was born in Barcelona, on 1986. He is studying the last course of telecommunication engineering at Universitat Politecnica de Catalunya. He is also working in the Theory of Signal and Communications Department as a teaching assistant in mobile communications.

Guillem Calduch was born in Barcelona, on 1986. He has studied telecommunication engineering at Universitat Politecnica de Catalunya. Currently he is working on his Master Thesis about Android operating system and development of applications for this OS.