Lecture Notes for Advanced Web Security 2017

Part 3 — Electronic Voting

Martin Hell

1 Introduction

Elections have been used for a long time to select e.g., representatives in parliaments or presidents. They can also be used in a much smaller scale to select board members in companies and organizations and even to decide upon specific questions within a small group. They form one of the cornerstones in a democratic system, and it is thus very important that the process of the election conforms to a given set of rules. The exact rules can be situation specific, but there are properties that can be seen as universal, such as correctness of the result.

It is important that the voting protocols are simple to use for the voters, otherwise the complexity of voting may stop people from voting. Electronic voting can potentially simplify the voting procedure since people might be allowed to vote from home, but it can also make it more difficult if it requires several steps of interaction with the voting system in order to preserve certain properties. Privacy is one such property, but there are several others as well.

If voters can cast their ballots on their home computer, using the Internet, it is possible to attract more voters, but it can also simplify coercion, i.e., to force someone to vote in a certain way using e.g., threats and it simplifies buying and selling of votes. These are problems that should be addressed in a secure voting system.

These lecture notes will give an introduction to electronic voting systems. None of the presented schemes are complete, but the examples instead serves as a general overview of the different possible approaches and the cryptographic tools that are needed to design and realize electronic voting protocols.

2 Electronic Voting

Electronic voting is a term that refers to at least two different things. It can refer to a voting system where some electronic device is used to receive and count the votes. Well known examples are Direct-Recording Electronic (DRE) machines. It also refers to Internet voting, where voters can use a home computer or possibly a smartphone in order to cast a vote.

There are three main phases of an election.

- **Voter Registration Phase** All eligible voters are registered as voters.
- **Voting Phase** All registered voters are allowed to cast their vote.
- **Tallying Phase** The votes are counted to obtain the final result.

The exact mechanisms in the three phases are very situation dependent.

2.1 Properties of Voting Schemes

There are many properties that should be considered in a secure voting system.

Privacy/Anonymity. It should be impossible for anyone to extract any information about someone else's vote. Exceptions to this are of course the information that can be extracted using the final result and the information that can be extracted using a voter's own ballot. The same should hold for groups, i.e., it should not be possible for a group of users to extract information about some voter's ballot.

Correctness. The result of the election matches the intention of the voters. This is also related to robustness and fault tolerance.

Verifiability. The verifiability property includes the possibility for different parties to verify the voting system. It can be divided into two parts.

- **Individual Verifiability** It should be possible for each voter to ensure that their vote was recorded as intended and included in the computation of the final result.
- **Universal Verifiability** It should be possible for a third party (i.e., anyone) to ensure that all votes have been included in the computation of the final result and that the election was properly performed.

Universal verifiability allows interested third parties to verify the result of the election. The voters themselves are in this case not responsible for checking the result since it can be delegated to organizations interested in verifying democratic procedures. There are other definitions and distinctions between different types of verifiability, but the above will be sufficient here.

Voter Eligibility. Only voters that are allowed to vote can vote. At the same time, it is important that eligible voters are not refused to vote. This means that the voters must be identified, but the identification process can not be such that it rules out some group of people because they can not acquire the proper identification credentials.

One-Voter-One-Vote. It should not be possible to vote twice. More specifically, it should not be possible for one voter to cast two votes that are both

included in the computation of the final result. Some systems may allow voters to vote several times with the result that only one vote (typically the last one) is counted. This is related to coercion-resistance (see below).

Receipt-Freeness. It should not be possible for a voter to prove how he/she votes. If some kind of receipt for the vote can be produced, it allows people to sell their vote.

Coercion-Resistance. It should not be possible to coerce someone to vote in a particular way. This is related to receipt-freeness, which does not necessarily imply coercion resistance since coercion can be accomplished even without a receipt. Just the fact that it is possible to vote on a computer from home makes coercion possible. One way to enable coercion resistance is to allow voters to cast several votes, with only the last one being counted. Then, even if someone is coerced into voting for a particular candidate, the voter always has the option of making that vote void by voting again.

Robustness/Fault Tolerance. Even if some parts of the system fail, e.g., equipment breaks down, the correct result should still be possible to obtain. The correct result should also be provided even if some parts of the system try to cheat or behave erroneously. At a minimum, the erroneous behaviour should be detected by the system. Similarly, privacy should be maintained even if a small set of authorities tries to break it.

Fairness. No partial results should be disclosed before the end of the voting procedure. As an example, if the tallying is divided into several districts, the result from one district must not be known before the end of the election process since this might influence remaining voters. This means that the privacy property should apply also to the tallier.

The above properties are security related. Other properties that are needed in a practical environment are the following.

- It should be easy to vote, meaning that the voter should be able to cast the vote and then be finished.
- Voting should be optional.

Privacy can be considered the most important property, and the main property that makes secure voting schemes difficult to implement. If privacy is not needed anyone could just log on to some system using an electronic ID and post their vote for anyone else to see it.

The type of election will influence the voting scheme. The most common elections are *yes/no voting* and *1-out-of-L voting*. In the first case, there are only two options and the voter should choose one. Swedish examples are the 1994 EU membership election and the 2003 Euro election. (Actually, yes/no voting does not include the possibility of voting null, so the examples are not

completely true.) In 1-out-of-L voting, the voters choose one option out of L possible. Other types are of course imaginable, such as *K-out-of-L voting* where voters choose K options out of a set of L options, or *K-out-of-L ordered voting* where the K chosen options are ordered, or *write-in voting* where the votes are text strings.

3 Tools

This section will give an overview of some tools that are used to realize electronic voting systems. For a more comprehensive treatment, refer to e.g., [7].

3.1 Bulletin Board

A bulletin board can be seen as a broadcast channel with memory. Anyone can write to their own section of the bulletin board and anyone can read everything that is posted on the board. However, it is not possible to delete or modify anything on the board. Typically, participants in the voting scheme can write to the board, but anyone, even third parties, can read the board. It is used to provide (universal) verifiability of a voting protocol. Being able to only write to one's own section on the board require some kind of access control. Thus, it is assumed that it is not possible to anonymously write to the bulletin board.

3.2 ElGamal Encryption

ElGamal encryption is an asymmetric cipher based on the difficulty of computing the discrete logarithm, i.e., knowing y, g and q it is difficult to find x such that $y = g^x \bmod q$ where q is a large prime.

The key generation consists of choosing a multiplicative group G of integers modulo a prime q with generator g. (Actually, q should be chosen as $q = 2p+1$, but that level of detail is out of scope here.) A random $x \in \{1, \ldots, q-2\}$ is chosen as the private key and the public key is given by $y = g^x \bmod q$, together with q and g.

The ciphertext consists of two values. When encrypting a message $m \in G$ using the public key, a random value r is first chosen. Then encryption of m using the chosen r is given by

$$E(m, r) = (a, b) = (g^r, m \cdot y^r)$$

Decryption is performed by computing $m = b/a^x$. This will give the correct message since

$$\frac{b}{a^x} = \frac{m \cdot g^{xr}}{g^{xr}} = m \bmod q.$$

ElGamal encryption is used in several electronic voting systems. One property of ElGamal encryption is that it is homomorphic. A homomorphic encryption system allows an operation to be carried out on the ciphertext values, which will

result in an encrypted value of an operation on the plaintext. More specifically, an encryption scheme E is said to be homomorphic if

$$E(m_1) * E(m_2) = E(m_1 *' m_2)$$

for some operators $*$ and $*'$. It is easy to see that the homomorphic property of ElGamal is given by

$$E(m_1, r_1)E(m_2, r_2) = E(m_1 m_2, r_1 + r_2)$$

since

$$E(m_1, r_1)E(m_2, r_2) = (a_1 a_2, b_1 b_2) = (g^{r_1+r_2}, m_1 m_2 y^{r_1+r_2}) = E(m_1 m_2, r_1 + r_2),$$

i.e., $*$ and $*'$ are both the multiplication operator. This means that by multiplying two encrypted messages, the result is the encryption of the product of the messages. Plain RSA also has this property, but falls short since the padding in RSA will destroy the homomorphic property.

Homomorphic encryption has lately been a very hot research topic. The research has been focused on systems that are called *fully homomorphic*, meaning that there are two operations that fullfill the requirements. Fully homomorphic schemes are e.g., very useful in cloud computing.

3.3 Zero-Knowledge Proofs

The idea of a zero-knowledge proof is to allow a prover to prove that she knows a secret without revealing any information about the secret itself. There are two parties involved, namely the prover, often referred to as Peggy, and the verifier, called Victor.

A typical, and easy to understand, example is that of proving knowledge of graph isomorphism. Two graphs, G_1 and G_2, are said to be isomorphic if it is possible to relabel the vertices of one graph such that the two graphs become equal. The isomorphism is thus a permutation of the vertices in one graph. It is in general considered difficult to determine if G_1 and G_2 are isomorphic, but it is very easy to determine if a permutation applied to G_1 produces G_2. The secret that Peggy knows is the actual permutation $\phi : G_1 \rightarrow G_2$ and she wants to prove that she knows this permutation without revealing it to Victor. This is done as follows. Peggy takes the graph G_2 and permutes the vertices to produce a new graph H. This permutation is called γ. Now Peggy knows the isomorphism between all graphs.

$$\begin{aligned} \phi &: G_1 \rightarrow G_2 \\ \gamma &: G_2 \rightarrow H \\ \gamma \circ \phi &: G_1 \rightarrow H \end{aligned}$$

but someone that does not know ϕ can only choose to construct H such that it is isomorphic to G_1 or isomorphic to G_2. Victor randomly picks one of γ

and $\gamma \circ \phi$ and asks Peggy to reveal this isomorphism. If Peggy knows ϕ, she will always be able to provide the requested isomorphism, but if she does not know ϕ there is only a probability $1/2$ that she can provide it, namely if she chose to construct the isomorphism that Victor asked for. The protocol can be repeated an arbitrary number of times in order to make the probability of cheating arbitrarily small.

Three main properties are related to these types of proofs.

- Completeness. If Peggy knows the secret, Victor will always accept her proof.
- Soundness. If Peggy does not know the secret, Victor will accept the proof with very small probability.
- Zero-Knowledge. By the end of the proof, Victor should have learnt nothing about the secret.

The zero-knowledge property is related to the existence of a simulator that simulates the steps of the proof. If it is possible for Victor to produce a (faked) transcript of the communication in the proof, and this transcript can not be distinguished from an actual transcript, then Victor can not have learnt anything during the proof.

Zero-Knowledge proofs are widely used in electronic voting in order for participants to e.g., prove that an encrypted vote is valid without revealing the vote, and to prove that certain parts of the system is behaving correctly without revealing certain private parameters.

A limitation is that the proofs are interactive by nature so it will require two-way communication in the verification step. This can be avoided by using a nice trick due to Fiat and Shamir [3]. The idea is to let the challenge provided by Victor be a function of some predetermined parameters. If the function inputs can not be fully controlled by Peggy, but the result can be computed, then this can be used as challenge, allowing Peggy to attach a proof together with the public values, e.g., an encrypted vote.

3.4 Secret Sharing

Secret sharing is a way of sharing a secret in such a way that several parties need to be involved in order to reconstruct it. A (t, n) threshold scheme is a secret sharing scheme with n parties, such that at least t parties need to cooperate in order to reconstruct the secret. Several such schemes have been proposed. The first, and probably the most well known, is called Shamir secret sharing [6]. It consists of a trusted dealer that knows the secret, and n participants $P_i, 1 \leq i \leq n$. The trusted dealer creates a polynomial

$$f(z) = x + \alpha_1 z + \alpha_2 z^2 \ldots + \alpha_{t-1} z^{t-1},$$

where $f(0) = x$ is the secret to be shared. The dealer creates the share for P_i by computing $f(i)$. Thus, each participant is given one evaluation of a function of degree $t-1$. Presumably, the trusted dealer destroys the secret at this point.

Using Lagrange interpolation, t out of the n participants are able to reconstruct the degree $t-1$ polynomial and thus find $f(0)$, which is the secret. Using $< t$ shares, no information about $f(0)$ can be obtained. A somewhat simplified variant of Lagrange interpolation can be used since the complete polynomial is not needed, only the point at $f(0)$. Let Λ be a set with t participants, then

$$x = f(0) = \sum_{i \in \Lambda} f(i) \prod_{j \in \Lambda \setminus \{i\}} \frac{j}{j-i}.$$

This is a nice and useful technique in situations where we do not want to rely on one single authority to behave correctly. However, it still requires one single authority in the first step when the polynomial is constructed. This authority will know the secret.

An extension to the secret sharing scheme above, and in the public key setting, is to let a group of n participants construct public/secret key pairs. Amazingly, it is possible for n participants to agree on a public key that anyone can use to encrypt, and then require t parties to be involved in the decryption with the private key, without anyone learning the private key in advance. The procedure is very similar to Shamir's secret sharing scheme above.

Each participant A_i constructs a private value x_i and computes a public value $y_i = g^{x_i} \bmod q$. If the discrete logarithm problem is difficult, it is also difficult to find the private value knowing the public value. All individual public values are multiplied to form the public key $y = y_1 y_2 \cdots y_n$ of the scheme. Thus, the private key is the sum of all private values since

$$y_1 y_2 \cdots y_n = g^{x_1 + x_2 + \ldots + x_n} \bmod q.$$

The goal is now to define a method for t participants to reconstruct $x = x_1 + x_2 + \ldots + x_n$. This is achieved as follows.

1. Each participant A_i forms a degree $t-1$ polynomial $f_i(z) = x_i + \alpha_{i,1} z + \ldots + \alpha_{i,t-1} z^{t-1}$.

2. A_i computes share $f_i(j)$ for participant j. Each other participant receives their own share, i.e., A_i sends $f_i(j)$ to P_j. These are sent encrypted to each participant, using some encryption scheme.

Consider the polynomial which is the sum of all privately formed polynomials,

$$\begin{aligned} f(z) &= f_1(z) + f_2(z) + \ldots + f_n(z) && (1) \\ &= x + \sum_{i=1}^{n} \alpha_{i,1} z + \sum_{i=1}^{n} \alpha_{i,2} z^2 + \ldots + \sum_{i=1}^{n} \alpha_{i,t-1} z^{t-1} && (2) \end{aligned}$$

Participant A_i can compute one point on this polynomial by summing up the shares received from the other participants, together with $f_i(i)$, which participants can compute themselves,

$$f(i) = f_1(i) + f_2(i) + \ldots + f_n(i).$$

Since all participants have received one point on this degree $t-1$ polynomial, t participants can recover x in the same way as before.

One extra feature in this scheme is that the shares can be verified, i.e., participant A_j can verify that the share received from A_i is a point on the polynomial selected by A_i. This is achieved by letting P_i publish (publicly) $F_{i,k} = g^{\alpha_{i,k}} \bmod q, 1 \leq j \leq t-1$. Then the value $f_i(j)$ can be verified by checking that

$$g^{f_i(j)} = \prod_{k=0}^{t-1} F_{i,k}^{j^k} \tag{3}$$

where $F_{i,0} = y_i = g^{x_i}$ is the public value that has already been shared. By verifying all shares received from the other participants, A_j can know that all shares will sum up to a point on the polynomial $f(z)$.

A variant of the secret sharing protocol above will be used when looking at electronic voting protocols based on homomorphic encryption.

4 Electronic Voting Schemes

There are a large number of proposals for electronic voting schemes. The examples given in these lecture notes only provide an introduction to electronic voting in order to give an overview of how electronic voting could be realized and the cryptographic primitives that are, or can be, involved. It is by no means a comprehensive overview.

In order to combine the privacy and verifiability properties, there are basically two different strategies.

- The vote is posted on the bulletin in clear text, but the person casting the vote is anonymous.
- The vote is posted on the bulletin board in encrypted form, but the person is not anonymous.

Both these strategies present challenges that need to be solved. In the first case, if the user is anonymous, how can we know that he/she is entitled to vote and how can we check that he/she does not double vote? In the second case, how can we count the votes if they are encrypted? Someone needs to decrypt the vote, but we can not afford to lose the privacy property. The tools presented in the previous section will be used to solve these, and other, problems.

In the literature, there are three main types of electronic voting schemes.

- Schemes based on mix networks
- Schemes based on blind signatures
- Schemes based on homomorphic encryption

Sometimes, the names are different. Mix networks can be generalized to verifiable shuffles, a name which not only reflects the privacy property, but also the universal verifiability property since anyone should be able to verify that the Mixes are behaving correctly.

4.1 Using a Mix Network

An electronic voting system must hide the connection between voters and the contents of their votes. The purpose of anonymous communication can be seen as a generalization of this, and using techniques from anonymization is a natural starting point when creating an electronic voting scheme. Consider the Chaum Mixes discussed on a previous lecture. These can be used to enable anonymous communication and to realize an anonymous channel. In [1], Chaum in addition to secure electronic email, also sketches a way to realize electronic elections. The idea is to construct a publicly readable list of public keys. A public key PK will be used as a pseudonym for the user. A digital signature of a message M can be computed using the corresponding private key SK. This will be denoted $\sigma_{SK}(M)$ or just $\sigma(M)$. (Quite often, the term secret key is used instead of private key in order to distinguish the abbreviations.) The list can be constructed by using a Mix network. Let K_i be the public key for Mix number i.

$$K_n(R_n, K_{n-1}(\ldots, K_2(R_2, K_1(R_1, PK))\ldots)).$$

This is the registration phase. The first Mix (i.e., Mix with public key K_n) checks the real identity of the sender to verify that it is an eligible voter before forwarding the message. The last Mix will then output a batch of digital pseudonyms $(PK_1, PK_2, \ldots)$ for all eligible voters, which will be posted publicly. Upon voting each user sends

$$K_n(R_n, K_{n-1}(\ldots, K_2(R_2, K_1(R_1, PK, V, \sigma(V)))\ldots))$$

through the Mixes and the last Mix can post $PK, V, \sigma(V)$ publicly. Again, the first Mix must check the identity of the voter. Note that it is assumed that anyone can listen to the traffic entering and leaving the Mixes. Thus, it is more natural to think of the communication channel as a bulletin board. All messages are written on a publicly readable bulletin board, i.e., input and output of each Mix, both in registration and voting phase. The important part is that Mixes secretly permute the list of messages. The final Mix will post the pseudonyms together with the votes. Now each voter can check that each pseudonym has voted and it can also check the vote and verify the result.

This scheme enables anonymity, which is important for elections. However, there are some important problems with this solution that motivated further research and improvements.

- If a single Mix is corrupt, e.g., it changes the encryption, then there will be an error in the posted result in the end. The voter may of course complain that something went wrong since voters can verify their own votes, but

then all votes have already been posted in the clear. A re-election would most likely be affected by this. Thus, the fairness property does not hold.

- Receipt-freeness does not hold since a voter can just show the private key to prove how the vote was cast.
- Third parties do not know that the Mixes are behaving correctly and produce the correct result. Thus the scheme is not universally verifiable (even though everyone can *count* and *see* the votes).

In order to enable universal verifiability the Mixes need to prove that they are behaving correctly, i.e., that they shuffle the ciphertext each time they have decrypted their own part of it. At the same time, the proof should not give away any information about the permutation, i.e., it is good if it is a zero-knowledge proof. They also need to prove that they have correctly decrypted the ciphertexts. Much research has been put into these questions, trying to make this as efficient as possible, not only the proofs but also the verification of the proofs.

4.2 Using Blind Signatures

Another variant of realizing electronic voting is to use blind signatures together with a commitment scheme (see lecture). These provide a straightforward way of allowing voters to make sure that their vote has been included in the final tally, before the vote is made public. They also clearly separate the identification of voters and the process of sending the ballot to an authority. One simple protocol based on blind signatures and bit commitment was described in [4] and the procedure below closely follows that description. Let $\xi(v, k)$ be a commitment to the value v and let the blinding of the message x be given by $\chi(x, r)$. Thus, using a signature on $\chi(x, r)$ it is possible to extract a signature on x if r is known.

The preparation and administration phase is given as follows.

- Voter V_i commits to a vote by computing $x_i = \xi(v_i, k_i)$. This will be the voter's ballot.
- The voter computes a blinded value of the commitment, $e_i = \chi(x_i, r_i)$, and signs this value using the private key $s_i = \sigma_{V_i}(e_i)$.
- An administrator A verifies the signature, identifies the voter, checks that the voter is allowed to vote, and has not applied for voting before. If everything is ok, A signs the blinded commitment, $d_i = \sigma_A(e_i)$.
- A publishes a list of $\langle ID_i, e_i, s_i \rangle$ on a bulletin board.

At this point, knowing the signature on e_i, and the value r_i, the voter can compute A's signature on the ballot x_i, $y_i = \sigma_A(x_i)$. Moreover, everyone can see a list of accepted voters, together with their blinded commitments. If one voter is not included in the list, the voter can complain. It is not possible for

voters to change their mind and no one can see anyone else's vote due to the commitment.

The next step in the protocol is the voting and collecting phase.

- Each voter sends the signed commitment to the Counter C using an anonymous channel. Note that it is signed by the administrator, not the voter. The signature is extracted knowing r_i.
- C verifies the signature and publishes l_i, x_i, y_i on the bulletin board, where l_i is just an integer identifying the ballot.

At this point everyone can compare the two lists, and if they do not match, releasing the value of r_i it is possible for everyone to verify which voter's ballot was not received. If there are no complaints, the protocol ends with the opening and counting phase.

- Voter V_i sends (l_i, k_i) through an anonymous communication channel, where l_i is the number identifying V_i's ballot. Using k_i it is possible to open the commitment and retrieve the vote identified by l_i.

Two important properties can be seen in this voting protocol.

- Even if the administrator and the counter cooperate, the privacy is still ensured due to the blind signature.
- It is not possible to extract any result before all voters have had the chance to verify that their own vote has been correctly registered. This was a problem with the original scheme described by Chaum in [1] (but it is not a problem with later Mix-based or verifiable shuffle-based schemes).

The protocol does not have the property of receipt-freeness. It is possible to add this property to the protocol. This was shown in [5], but it requires the existence of an *untappable* anonymous channel. This requirement is much stronger than just requiring an anonymous channel. Note that a channel does not become untappable just because the communication is encrypted. Not even encrypted communication is allowed to be read by an adversary since this would remove the receipt-freeness.

4.3 Using Homomorphic Encryption

A third major variant of electronic voting are schemes based on homomorphic encryption. The example given here will use the homomorphic property of ElGamal. Recall that this is given by

$$E(m_1, r_1)E(m_2, r_2) = E(m_1 m_2, r_1 + r_2).$$

The idea is to perform an operation on the ciphertexts submitted by each voter, and then decrypt the result. As was seen earlier, the homomorphic property of ElGamal allows ciphertexts to be multiplied resulting in an encryption of the

product of the two plaintexts. This is not very useful since votes are *summed* up to get the final result. Instead, consider a modification of ElGamal where the ciphertext is computed as $(a, b) = (g^r, w^m y^r)$ (instead of $(a, b) = (g^r, my^r)$). Then the homomorphic property will be

$$E(m_1, r_1)E(m_2, r_2) = (a_1 a_2, b_1 b_2) = (g^{r_1+r_2}, w^{m_1+m_2} y^{r_1+r_2}) = E(m_1+m_2, r_1+r_2).$$

which is exactly what we want. This scheme has a major drawback, namely that it is not the actual message m that is decrypted but instead $w^m \bmod q$. This means that in order to reconstruct m, a discrete logarithm has to be solved, which is known to be difficult.

There is a variety of electronic voting schemes based on homomorphic encryption. A summary of the example given in this section is as follows.

1. A user encrypts the vote using a homomorphic threshold scheme. The encrypted vote is published on a bulletin board so everyone can see who has voted.
2. The voter proves that the vote is valid.
3. All encrypted votes are multiplied.
4. A set of authorities cooperate to decrypt the sum or product.
5. Everyone can verify that the product of the encrypted votes is indeed a valid encryption of the final result. This gives universal verifiability.

These steps do not fit all schemes based on homomorphic encryption but they still capture the main ideas. One variant is to share the vote between the authorities instead of sharing the decryption key. Then each authority has its own decryption key but can only decrypt a part of the vote. The different parts are then combined in a way that is very similar to the secret sharing method described earlier.

For simplicity, consider a yes/no voting scheme, i.e., a vote must take one out of two values (which of course do not have to be literal yes or no). The values will be mapped to $\{-1, 1\}$ so these are the actual votes that are cast. A threshold scheme for public key encryption will be used to encrypt votes and to decrypt the product of the votes.

4.3.1 Threshold Encryption Scheme

If there is only one authority, then the use of homomorphic encryption to provide privacy is rather useless. Since the authority can decrypt the product of the ciphertexts using its private key, it can also decrypt any individual vote using the private key. Thus, there is no privacy since the authority, if it wants, can find out how each voter voted. Instead, consider the case when we do not necessarily trust the authority, or when it can be allowed to malfunction. The solution then is to use several authorities and a threshold scheme. Then, even if one or a few

authorities are corrupt they will not be able to decrypt the votes. It will require at least t cooperating authorities to perform the decryption.

The threshold scheme will use ElGamal encryption and is similar to the secret sharing scheme described in Section 3.4. The system has a set of n authorities $(A_1, A_2, \ldots, A_n)$ that each has one private value x_i and one public value $y_i = g^{x_i} \bmod q$. As before, authority A_i constructs a polynomial $f_i(z)$ with $f_i(0) = x_i$. Each other authority $A_j, j \neq i$ is given one evaluation of this polynomial, namely $f_i(j)$. The correctness of these can be verified using (3). Summing all received evaluations, A_j will get one evaluation of $f(z) = \sum f_i(z)$, i.e., a share of the secret x. These are denoted s_i for authority A_i. Each authority then publishes $h_i = g^{s_i} \bmod q$ as a commitment to the value s_i.

The public key is given by $y_1 y_2 \cdots y_n$. The public key can be used by anyone to encrypt a message m as

$$E(m, r) = (a, b) = (g^r, my^r),$$

where r is a random number. The encrypted message can only be decrypted by t co-operating authorities. In the scheme presented in Section 3.4, the authorities co-operated to recover $x = x_1 + x_2 + \ldots + x_n$. In order to make the threshold scheme suitable for an electronic voting scheme, the following properties will be added.

- The message will be decrypted without anyone actually learning the private key x.
- Authorities will be able to prove that they are behaving correctly and *anyone* will be able to verify that the message has been correctly decrypted (still without anyone learning the private key).

These two properties are achieved as follows. Each authority publishes $u_i = a^{s_i} \bmod q$ (note that the a is taken from the ciphertext). Now, each authority has published two public values, namely

$$h_i = g^{s_i} \bmod q \tag{4}$$
$$u_i = a^{s_i} \bmod q. \tag{5}$$

The values g and a are of course also public. Let Λ be a set of t authorities. Now, m is decrypted by computing

$$m = \frac{b}{\prod_{i \in \Lambda} u_i^{\lambda_{i,\Lambda}}} \left(= \frac{b}{g^{r \sum_{i \in \Lambda} s_i \lambda_{i,\Lambda}}} = \frac{b}{y^r} \right)$$

where

$$\lambda_{i,\Lambda} = \prod_{j \in \Lambda \setminus \{i\}} \frac{j}{j - i}.$$

In order for a decrypting party to know that it is the correct value that is decrypted, it must know that the values u_i are in fact $a^{s_i} = g^{r s_i}$. This is

ensured by letting each authority prove that it knows the value s_i and to prove that s_i is the same in both equations, i.e., to prove that it knows s_i and that

$$\log_g h_i = \log_a u_i.$$

If this can be proven, anyone (that sees the proof) can be assured that the decrypted value *is* the original message. The proof is summarized in Figure 1.

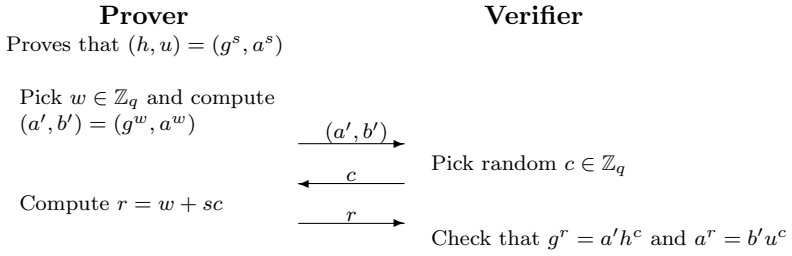

Figure 1: Proof that $\log_g h_i = \log_a u_i$.

It is not at all obvious why the proof is constructed exactly like this, and it is outside the scope of the course to cover the details. It can be noted that the proof is an *honest verifier zero-knowledge proof* since the existence of a simulator that is indistinguishable from a valid transcript requires the verifier to select c randomly. (It can not be based on g^w or a^w.) It further has a property called special soundness, which means that with two protocol runs with transcripts (a', b', c, r) and (a', b', c', r'), it is possible to extract the secret s.

The prover proves her knowledge of s by being able to compute r such that the verification holds. Moreover, the existence of a simulator shows that the proof is zero-knowledge (for an honest verifier). The prover also shows that $\log_g h_i = \log_a u_i$ as follows. Assume that $h = g^s$ and $u = a^{\widehat{s}}$, and let the prover choose $a' = g^w$ and $b' = a^{\widehat{w}}$. Then the verification will hold if

$$w + sc = \widehat{w} + \widehat{s}c \Rightarrow c = \frac{\widehat{w} - w}{s - \widehat{s}},$$

which only holds with small probability.

It must be noted that certain restrictions apply when choosing the parameters in the threshold encryption scheme. ElGamal over the integers modulo q is done in a group of order $q - 1$. Thus, if a generator g of this group is used it will have order $q - 1$. Since $g^{ord(g)} = g^{q-1} = 1 \bmod q$, the polynomials and the Lagrange interpolation then has to be computed modulo $q - 1$, i.e., everything in the exponent is computed modulo $q - 1$. This is a problem because not all integers will have inverses modulo $q - 1$ (since this is an even number). What can be done instead is to let g generate a subgroup of prime order p. Then $g^p = 1 \bmod q$. A simple way of doing this, making sure that $ord(g) = p$ is large, is to find a prime q such that $q - 1 = 2p$. All elements in g then have

order $1, 2, p$ or $2p$. If $\hat{g}$ is a generator of the multiplicative group modulo q, then $g = \hat{g}^2$ has order p. Thus, ElGamal can be computed modulo q, while Lagrange (the exponents) are computed modulo p. As an example $q = 1523$, $p = 761$ and $g = 4$ can be used as (toy) parameters, since $1522 = 2 \cdot 761$ and 4 has order 761 in the multiplicative group of integers modulo q.

4.3.2 Applying this to Electronic Voting

The homomorphic property of ElGamal together with the threshold scheme can be used to construct an electronic voting scheme. Each user uses the public key y in the threshold encryption scheme to encrypt a vote. The values g, w and q are considered public. Since the voting scheme requires the homomorphic operator to be addition in the message space, each voter V_i encrypts the vote $v_i \in \{-1, 1\}$ as

$$E(v_i, r_i) = (a_i, b_i) = (g^{r_i}, w^{v_i} y^{r_i}).$$

Then the voter uses a non-interactive proof to prove that the vote is either -1 or 1 and nothing else. This proof will not be given here, but one can be found in [2]. The encrypted vote and the proof are written on a bulletin board and everyone can verify who voted and that the vote is one of the allowed values. At the end of the voting phase, all encrypted votes are multiplied. If there are m voters, then

$$\prod_{i=1}^{m} E(v_i, r_i) = (\prod_{i=1}^{m} a_i, \prod_{i=1}^{m} b_i) = (g^{\sum_{i=1}^{m} r_i}, w^{\sum_{i=1}^{m} v_i} y^{\sum_{i=1}^{m} r_i}) = E(w^{\sum_{i=1}^{m} v_i}, \sum_{i=1}^{m} r_i).$$

Using t values of $u_i = a^{s_i}$, it is possible to decrypt the ciphertext and recover

$$w^{\sum_{i=1}^{m} v_i},$$

where the exponent is the result of the election. If the exponent is negative, the number of -1-votes are in majority, and if it is positive there is a majority of 1-votes. Recovering the result requires solving a discrete logarithm, but if the number of voters is moderate, it is possible to exhaustively search the result.

This voting scheme is robust since even if some authorities behave erroneously, this will be discovered when they prove the correctness of u_i, and other values will be used in the decryption. As long as at least t authorities behave correctly, the correct result can be computed. It has universal verifiability since anyone can check the proofs and the correctness of the result. However, it does not have receipt-freeness.

Improvements to this scheme are e.g., more efficient proofs and other encryption methods. In particular, replacing ElGamal with a cryptosystem that has the appropriate operations in the homomorphic properties is an important improvement. The Paillier encryption system is a notable alternative that is homomorphic with addition as operation in the message space and multiplication in the ciphertext space. Thus, it has the same homomorphic property as the one used in the electronic voting scheme above, without the need for exhaustive search in order to solve a discrete logarithm.

References

[1] D. Chaum. Untraceable electronic mail, return addresses and digital pseudonyms. *Communications of the ACM*, 24:84–88, February 1981.

[2] R. Cramer, R Gennaro, and B Schoenmakers. A secure and optimally efficient multi-authority election scheme. In W. Fumy, editor, *Advances in Cryptology—EUROCRYPT'97*, volume 1233 of *Lecture Notes in Computer Science*, pages 103–118. Springer, 1997.

[3] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In A. M. Odlyzko, editor, *CRYPTO86*, volume 263 of *LNCS*, pages 186–194. Springer, 1986.

[4] A. Fujioka, T. Okamoto, and K. Ohta. A practical secret voting scheme for large scale elections. In J. Seberry and Y Zheng, editors, *Advances in Cryptology - AUSCRYPT '92*, volume 718 of *Lecture Notes in Computer Science*, pages 244–251. Springer, 1992.

[5] T. Okamoto. Receipt-free electronic voting schemes for large scale elections. In B Christianson, B. Crispo, T. Lomas, and M. Roe, editors, *Security Protocols Workshop*, volume 1361 of *Lecture Notes in Computer Science*, pages 25–35. Springer, 1997.

[6] A. Shamir. How to share a secret. *Communications of the ACM*, 22:612–613, 1979.

[7] N. Smart. Cryptography: An introduction, 2012. Available at: *http://www.cs.bris.ac.uk/~nigel/Crypto_Book/*.