Securing e-voting systems

Lorena Ronquillo

lron@demtech.dk

12/05/2015

DEMTECH
DEMOCRATIC TECHNOLOGY

This Lecture

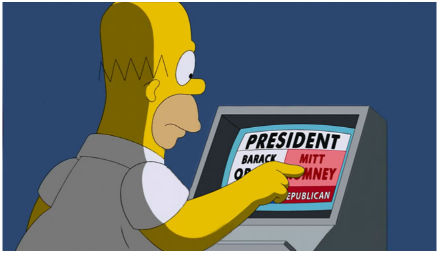
Introduction to e-voting

E-voting systems

Electronic voting (also known as **e-voting**) is voting using electronic systems to aid casting and counting votes.

We can identify two types:

- electronic voting machines located at polling stations (including direct-recording electronic voting systems, or **DRE**)
- remote voting, also called **Internet voting**.

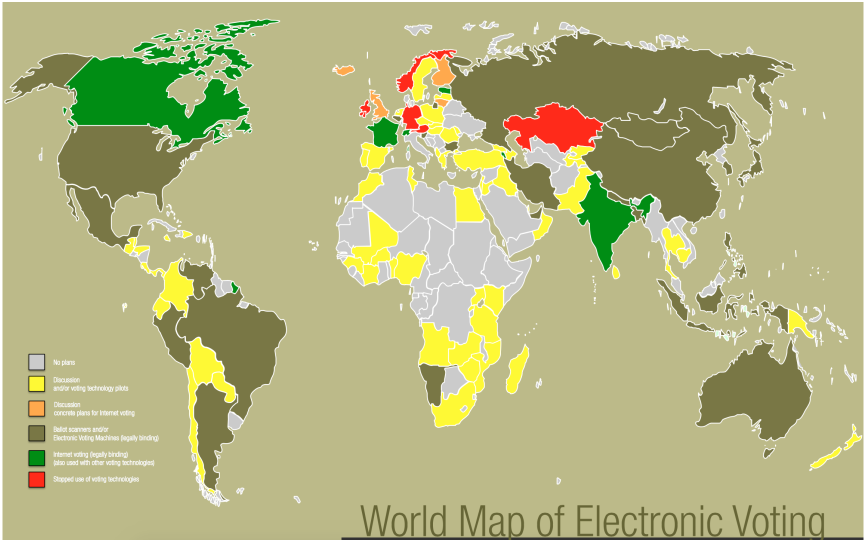


Countries using Internet voting (I)

Some countries using Internet voting are:

- France: for citizens living abroad.
- Norway: pilots in 2011 and 2013 for local and national elections. Discontinued.
- Estonia: for municipal and parliamentary elections since 2005.
- Switzerland: in several cantons.
- Australia (New South Wales, Victoria)
- Canada: municipal and provincial elections.

Countries using Internet voting (II)


No plans
Discussion and/or voting technology pilots
Discussion concrete plans for Internet voting
Ballot scanners and/or Electronic Voting Machines (legally binding)
Internet voting (legally binding) (also used with other voting technologies)
Stopped use of voting technologies
World Map of Electronic Voting

Source: e-voting.cc

Supporters and opponents arguments

☺ :

- Reducing costs of conducting an election or referendum
- Delivering election results **reliably** and more quickly
- Providing additional voting channels to increase voter turnout
- Increasing the number of elections more easily
- Improving access to the voting process to people with disabilities
- Bringing voting in line with new developments in society and increasing use of new technologies

Supporters and opponents arguments

☺ :

- Reducing costs of conducting an election or referendum
- Delivering election results **reliably** and more quickly
- Providing additional voting channels to increase voter turnout
- Increasing the number of elections more easily
- Improving access to the voting process to people with disabilities
- Bringing voting in line with new developments in society and increasing use of new technologies

☹ :

- Risk of (large-scale) manipulation by (small group of) insiders
- More difficult to detect and identify the source of errors and technical malfunctions than with conventional procedures
- Need for additional voter education campaigns
- Risk of (loosing) public trust in the election/referendum process

Security requirements

- **Integrity**: the outcome should match voters intent.
    - **Individual verifiability**: the voter should be able to verify that
        - *cast as intended*
        - *recorded as cast*
        - *counted as recorded*
    - **Universal verifiability**: any interested party shoud be able to verify that the tally is correct.

Security requirements

- **Integrity**: the outcome should match voters intent.
  - **Individual verifiability**: the voter should be able to verify that
    - *cast as intended*
    - *recorded as cast*
    - *counted as recorded*
  - **Universal verifiability**: any interested party shoud be able to verify that the tally is correct.
- **Eligibility**: only legitimate voters can cast votes.

Security requirements

- **Integrity**: the outcome should match voters intent.
  - **Individual verifiability**: the voter should be able to verify that
    - *cast as intended*
    - *recorded as cast*
    - *counted as recorded*
  - **Universal verifiability**: any interested party shoud be able to verify that the tally is correct.
- **Eligibility**: only legitimate voters can cast votes.
- **Privacy**: nobody can figure out how a voter voted.

Security requirements

- **Integrity**: the outcome should match voters intent.
    - **Individual verifiability**: the voter should be able to verify that
        - *cast as intended*
        - *recorded as cast*
        - *counted as recorded*
    - **Universal verifiability**: any interested party shoud be able to verify that the tally is correct.
- **Eligibility**: only legitimate voters can cast votes.
- **Privacy**: nobody can figure out how a voter voted.
- **Receipt-freeness**: a voter should not be able to prove how she voted.

Security requirements

- **Integrity**: the outcome should match voters intent.
  - **Individual verifiability**: the voter should be able to verify that
    - *cast as intended*
    - *recorded as cast*
    - *counted as recorded*
  - **Universal verifiability**: any interested party shoud be able to verify that the tally is correct.
- **Eligibility**: only legitimate voters can cast votes.
- **Privacy**: nobody can figure out how a voter voted.
- **Receipt-freeness**: a voter should not be able to prove how she voted.
- **Coercion-resistance**: no party should be able to force another party to vote in a certain way or abstain from voting.

Security requirements

- **Integrity**: the outcome should match voters intent.
  - **Individual verifiability**: the voter should be able to verify that
    - *cast as intended*
    - *recorded as cast*
    - *counted as recorded*
  - **Universal verifiability**: any interested party shoud be able to verify that the tally is correct.
- **Eligibility**: only legitimate voters can cast votes.
- **Privacy**: nobody can figure out how a voter voted.
- **Receipt-freeness**: a voter should not be able to prove how she voted.
- **Coercion-resistance**: no party should be able to force another party to vote in a certain way or abstain from voting.
- **Fairness**: no partial results should be known before the election is closed.

Security requirements

- **Integrity**: the outcome should match voters intent.
  - **Individual verifiability**: the voter should be able to verify that
    - *cast as intended*
    - *recorded as cast*
    - *counted as recorded*
  - **Universal verifiability**: any interested party shoud be able to verify that the tally is correct.
- **Eligibility**: only legitimate voters can cast votes.
- **Privacy**: nobody can figure out how a voter voted.
- **Receipt-freeness**: a voter should not be able to prove how she voted.
- **Coercion-resistance**: no party should be able to force another party to vote in a certain way or abstain from voting.
- **Fairness**: no partial results should be known before the election is closed.
- Availability, accessibility, etc.

Election verifiability vs. ballot privacy

Hand raising processes for voting are fully verifiable but...

Election verifiability vs. ballot privacy

Hand raising processes for voting are fully verifiable but...

... they are not private at all!

Election verifiability vs. ballot privacy

Hand raising processes for voting are fully verifiable but...

... they are not private at all!

Non-verifiable voting schemes might be fully private (using standard crypto and a completely trusted decryption and counting system) but...

Election verifiability vs. ballot privacy

Hand raising processes for voting are fully verifiable but...

... they are not private at all!

Non-verifiable voting schemes might be fully private (using standard crypto and a completely trusted decryption and counting system) but...

... there is no way for us to check that the tally is correct!

Election verifiability vs. ballot privacy

Hand raising processes for voting are fully verifiable but...

... they are not private at all!

Non-verifiable voting schemes might be fully private (using standard crypto and a completely trusted decryption and counting system) but...

... there is no way for us to check that the tally is correct!

Usually the voting process produces an audit trail (electronic, of paper, or both) for verifiability.

Election verifiability vs. ballot privacy

Hand raising processes for voting are fully verifiable but...

... they are not private at all!

Non-verifiable voting schemes might be fully private (using standard crypto and a completely trusted decryption and counting system) but...

... there is no way for us to check that the tally is correct!

Usually the voting process produces an audit trail (electronic, of paper, or both) for verifiability.

There is a **compromise between correctness and privacy** that needs to be made in the vast majority of the verifiable voting schemes.

Achieving privacy

E-voting uses public-key cryptography:

- public key: used by voters to encrypt their vote.
- private key: generated by the election authority and necessary to decrypt the encrypted votes and compute the final tally.

Many different encryption algorithms can be used, but the most common one is ElGamal.

Key storage

How to secure the secret key of an election?

Key storage

How to secure the secret key of an election?

- Encrypting the key: vicious cycle.

Key storage

How to secure the secret key of an election?

- Encrypting the key: vicious cycle.
- Replicating the key: insecure.

Key storage

How to secure the secret key of an election?

- Encrypting the key: vicious cycle.
- Replicating the key: insecure.

Problem: we don't want the whole privacy of the votes to rely on one single election authority!

Key storage

How to secure the secret key of an election?

- Encrypting the key: vicious cycle.
- Replicating the key: insecure.

Problem: we don't want the whole privacy of the votes to rely on one single election authority!

Idea: Distribute the key to a group of people, such that nobody by himself knows it.

Secret sharing

(k, n)-Secret sharing schemes

Also known as (k, n)-**threshold schemes**.

A dealer shares a secret key between n parties in such a way that

- Each party $i \in \{1, \ldots n\}$ receives a share.
- A group of any k participants can cooperate to reconstruct the secret from their shares.
- No group of less than k participants can get **any** information about the secret.

(k, n)-Secret sharing schemes

Also known as (k, n)-**threshold schemes**.

A dealer shares a secret key between n parties in such a way that

- Each party $i \in \{1, \ldots n\}$ receives a share.
- A group of any k participants can cooperate to reconstruct the secret from their shares.
- No group of less than k participants can get **any** information about the secret.

Example:

The president of a company has 3 shares, the prime minister has 2 shares, and other ministers have 1 share each. Then, by using a $(3, n)$-secret sharing scheme the secret key will be recovered by either of these group of people:

- the president
- the prime minister and another minister
- any three ministers

$(2,2)$-Secret sharing schemes

Let s be a secret from a group $(G, +)$. Dealer chooses at random $s_1 \xleftarrow{R} G$ and lets $s_2 = s - s_1$.

The two shares are s_1 and s_2.

$(2,2)$-Secret sharing schemes

Let s be a secret from a group $(G,+)$. Dealer chooses at random $s_1 \overset{R}{\leftarrow} G$ and lets $s_2 = s - s_1$.

The two shares are s_1 and s_2.

Then,

- Given s_1 and s_2 one can succesfully recover the secret $s = s_1 + s_2$.
- Given only s_1 (or only s_2), the other share is random.

Shamir's (k, n)-threshold scheme (I)

Dealing phase

Let s be a secret from some $\mathbb{Z}_p$, with p prime.

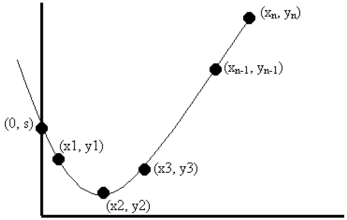
The dealer selects a random polynomial of degree $k - 1$

$$f(x) = f_0 + f_1 x + f_x x^2 + \cdots + f_{k-1} x^{k-1}$$

where

- coefficients $f_1, \ldots, f_{k-1}$ are selected at random from $\mathbb{Z}_p$
- $f_0 = s$

For $i \in \{1, \ldots, n\}$, the dealer distributes the share $s_i = (i, f(i))$ to party i.



Shamir's (k, n)-threshold scheme (II)

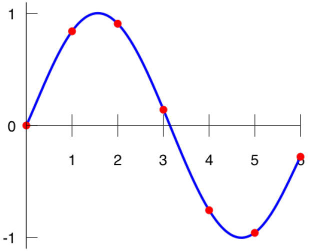
Reconstruction phase

The secret s can be reconstructed from every subset of k shares.
By the Lagrange formula, given k points (x_i, y_i), with $i = 1, \dots, k$,

$$f(x) = \sum_{i=1}^{k} y_i \prod_{j=1, j \neq i}^{k} \frac{x - x_j}{x_i - x_j} \pmod p$$

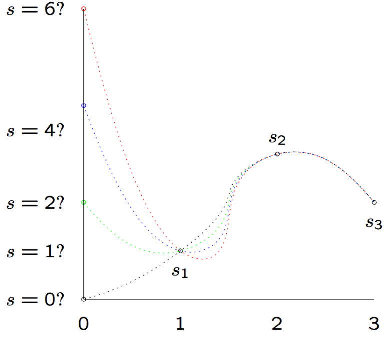
and thus

$$s = f(0) = \sum_{i=1}^{k} y_i \prod_{j=1, j \neq i}^{k} \frac{-x_j}{x_i - x_j} \pmod p.$$



Shamir's scheme: security

Given less than k shares, the secret s (point $(0, s)$ at the graph) can have any value.



Shamir's scheme: flexibility

- We can increase n and add new shares without affecting other shares.
- Existing shares can be removed without affecting other shares (the share needs to be really destroyed).
- It is possible to replace all the shares (or just k shares) without changing the secret by selecting a new polynomial $\hat{f}(x)$ and a new set of shares (**proactive security**).
- There exist a distributed version of Shamir's scheme that avoids having a dealer.

Now we have

- a way to keep the votes secret (encryption),
- a way to safely store the election secret key (secret sharing).

Now we have

- a way to keep the votes secret (encryption),
- a way to safely store the election secret key (secret sharing).

1. The election secret key is shared among a group of election authorities. The public key of the election is derived from the private key.
2. Voters encrypt their vote using the public key and post them in the bulletin board.
3. The election authorities join their shares and recover the secret key to decrypt every ciphertext and compute the tally.

Now we have

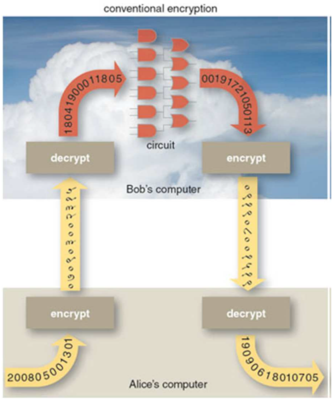
- a way to keep the votes secret (encryption),
- a way to safely store the election secret key (secret sharing).

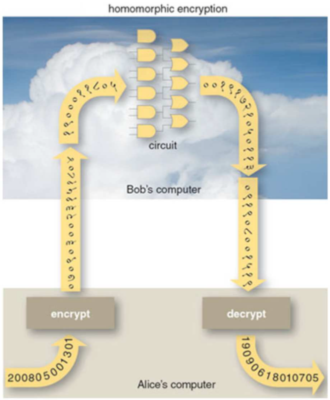
1. The election secret key is shared among a group of election authorities. The public key of the election is derived from the private key.
2. Voters encrypt their vote using the public key and post them in the bulletin board.
3. The election authorities join their shares and recover the secret key to decrypt every ciphertext and compute the tally.

We have to decrypt every ciphertext **one by one** before counting, can we do better?

Homomorphic encryption

What is homomorphic encryption?





Homomorphic encryption allows computing on data while it is encrypted rather than having to decrypt it first.

Refreshing ElGamal cryptosystem

Gen: Select a subgroup $G \subset \mathbb{Z}_p^*$ of order q, and a generator g of G. Choose $a \xleftarrow[R]{} \mathbb{Z}_q$.

- **Private key:** a
- **Public key:** $y = g^a$

Enc: To encrypt a message $m \in G$, we choose $b \xleftarrow[R]{} \mathbb{Z}_q$. The ciphertext is then

$$(c, d) = (g^b, m \cdot y^b).$$

Dec: To decrypt the ciphertext (c, d), compute

$$m = \frac{d}{(c)^a}.$$

Multiplicative homomorphism

ElGamal encryption has a multiplicative homomorphic property.

Multiplicative homomorphism

ElGamal encryption has a multiplicative homomorphic property.

Consider two encryptions (under the same public key y)

$$\begin{array}{ll} \text{encryption of } m_1 & (c_1, d_1) = (g^{b_1}, m_1 \cdot y^{b_1}), \\ \text{encryption of } m_2 & (c_2, d_2) = (g^{b_2}, m_2 \cdot y^{b_2}). \end{array}$$

Multiplicative homomorphism

ElGamal encryption has a multiplicative homomorphic property.

Consider two encryptions (under the same public key y)

$$\begin{aligned} &\text{encryption of } m_1 && (c_1, d_1) = (g^{b_1}, m_1 \cdot y^{b_1}), \\ &\text{encryption of } m_2 && (c_2, d_2) = (g^{b_2}, m_2 \cdot y^{b_2}). \end{aligned}$$

Multiplying these two ciphertexts we obtain

$$\begin{aligned} (c_1 \cdot c_2, d_1 \cdot d_2) =& (g^{b_1} \cdot g^{b_2}, m_1 \cdot m_2 \cdot y^{b_1} \cdot y^{b_2}) \\ =& (g^{b_1+b_2}, m_1 \cdot m_2 \cdot y^{b_1+b_2}) \qquad \text{encryption of } m_1 \cdot m_2 \end{aligned}$$

Additive Homomorphism

E-voting would benefit from an additive homomorphism.

$$\begin{tikzcd}
m_1 \arrow[d] & & m_2 \arrow[d] \\
E_{pk}(m_1) \arrow[dr] & & E_{pk}(m_2) \arrow[dl] \\
& \times \arrow[d] & \\
& E_{pk}(m_1 + m_2) &
\end{tikzcd}$$

Additive Homomorphism

E-voting would benefit from an additive homomorphism.

$$\begin{tikzcd}
m_1 \arrow[d] & & m_2 \arrow[d] \\
E_{pk}(m_1) \arrow[dr] & & E_{pk}(m_2) \arrow[dl] \\
& \times \arrow[d] & \\
& E_{pk}(m_1 + m_2) &
\end{tikzcd}$$

Solution: modify ElGamal. Put the plaintext in the exponent.

Exponential ElGamal

Instead of encrypting a message $m \in G$, we encrypt a message $g^m \in G$, where g is a generator in G.

Consider two encryptions

$$\begin{aligned} &\text{encryption of } g^{m_1} \qquad (c_1, d_1) = (g^{b_1}, g^{m_1} \cdot y^{b_1}), \\ &\text{encryption of } g^{m_2} \qquad (c_2, d_2) = (g^{b_2}, g^{m_2} \cdot y^{b_2}). \end{aligned}$$

Exponential ElGamal

Instead of encrypting a message $m \in G$, we encrypt a message $g^m \in G$, where g is a generator in G.

Consider two encryptions

$$\begin{aligned} &\text{encryption of } g^{m_1} \qquad (c_1, d_1) = (g^{b_1}, g^{m_1} \cdot y^{b_1}), \\ &\text{encryption of } g^{m_2} \qquad (c_2, d_2) = (g^{b_2}, g^{m_2} \cdot y^{b_2}). \end{aligned}$$

Multiplying these two ciphertexts we obtain

$$\begin{aligned} (c_1 \cdot c_2, d_1 \cdot d_2) =& (g^{b_1} \cdot g^{b_2}, g^{m_1} \cdot g^{m_2} \cdot y^{b_1} \cdot y^{b_2}) \\ =& (g^{b_1+b_2}, g^{m_1+m_2} \cdot y^{b_1+b_2}) \qquad \text{encryption of } g^{m_1+m_2} \end{aligned}$$

Exponential ElGamal

Instead of encrypting a message $m \in G$, we encrypt a message $g^m \in G$, where g is a generator in G.

Consider two encryptions

$$\begin{aligned} &\text{encryption of } g^{m_1} \qquad (c_1, d_1) = (g^{b_1}, g^{m_1} \cdot y^{b_1}), \\ &\text{encryption of } g^{m_2} \qquad (c_2, d_2) = (g^{b_2}, g^{m_2} \cdot y^{b_2}). \end{aligned}$$

Multiplying these two ciphertexts we obtain

$$\begin{aligned} (c_1 \cdot c_2, d_1 \cdot d_2) =& (g^{b_1} \cdot g^{b_2}, g^{m_1} \cdot g^{m_2} \cdot y^{b_1} \cdot y^{b_2}) \\ =& (g^{b_1+b_2}, g^{m_1+m_2} \cdot y^{b_1+b_2}) \qquad \text{encryption of } g^{m_1+m_2} \end{aligned}$$

If $m_1 + m_2$ are not too big, it is possible to efficiently solve the discrete logarithm of $g^{m_1+m_2}$ and thus obtain $m_1 + m_2$.

Homomorphic encryption and e-voting (I)

Example
Let y be the public key of an election.

We assume that each vote is a yes (✓) or no for each candidate, encoded by 1 and 0, respectively.

Suppose Alice, Bob and Oscar are running as candidates in an election. Only 5 people voted in the election, and the results are tabulated below.

	Oscar	Bob	Alice
voter 1			✓
voter 2		✓	
voter 3		✓	
voter 4			✓
voter 5	✓		

Homomorphic encryption and e-voting (II)

Casting a vote

Each voter posts to the bulletin board (BB) the following encrypted ballots:

BB	Oscar	Bob	Alice
voter 1	$(g^{r_{11}}, g^0 y^{r_{11}})$	$(g^{r_{12}}, g^0 y^{r_{12}})$	$(g^{r_{13}}, g^1 y^{r_{13}})$
voter 2	$(g^{r_{21}}, g^0 y^{r_{21}})$	$(g^{r_{22}}, g^1 y^{r_{22}})$	$(g^{r_{23}}, g^0 y^{r_{23}})$
voter 3	$(g^{r_{31}}, g^0 y^{r_{31}})$	$(g^{r_{32}}, g^1 y^{32})$	$(g^{r_{33}}, g^0 y^{r_{33}})$
voter 4	$(g^{r_{41}}, g^0 y^{r_{41}})$	$(g^{r_{42}}, g^0 y^{r_{42}})$	$(g^{r_{43}}, g^1 y^{r_{43}})$
voter 5	$(g^{r_{51}}, g^1 y^{r_{51}})$	$(g^{r_{52}}, g^0 y^{r_{52}})$	$(g^{r_{53}}, g^0 y^{r_{53}})$

Homomorphic encryption and e-voting (III)

Tally of the election

BB	Oscar	Bob	Alice
voter 1	$(g^{r_{11}}, g^0 y^{r_{11}})$	$(g^{r_{12}}, g^0 y^{r_{12}})$	$(g^{r_{13}}, g^1 y^{r_{13}})$
voter 2	$(g^{r_{21}}, g^0 y^{r_{21}})$	$(g^{r_{22}}, g^1 y^{r_{22}})$	$(g^{r_{23}}, g^0 y^{r_{23}})$
voter 3	$(g^{r_{31}}, g^0 y^{r_{31}})$	$(g^{r_{32}}, g^1 y^{32})$	$(g^{r_{33}}, g^0 y^{r_{33}})$
voter 4	$(g^{r_{41}}, g^0 y^{r_{41}})$	$(g^{r_{42}}, g^0 y^{r_{42}})$	$(g^{r_{43}}, g^1 y^{r_{43}})$
voter 5	$(g^{r_{51}}, g^1 y^{r_{51}})$	$(g^{r_{52}}, g^0 y^{r_{52}})$	$(g^{r_{53}}, g^0 y^{r_{53}})$

$\downarrow$ multiplying

$(g^{r}, g^1 y^{r})$	$(g^{r'}, g^2 y^{r'})$	$(g^{r''}, g^2 y^{r''})$

where $r = r_{11} + r_{21} + \ldots + r_{51}$

$$r' = r_{12} + r_{22} + \ldots + r_{52}$$

$$r'' = r_{13} + r_{23} + \ldots + r_{53}$$

Decrypting using the private key of the election, we obtain g^1, g^2, g^2.
These discrete logarithms are easy to compute, but can also be pre-computed before the election (lookup table).

1. The election secret key is shared among a group of election authorities. The public key of the election is derived from the private key.
2. Voters encrypt their vote using the public key and post them in the bulletin board.
3. **Encrypted votes are multiplied (by candidate).**
4. The election authorities join their shares and recover the secret key to decrypt **only one ciphertext per candidate** and compute the tally.

1. The election secret key is shared among a group of election authorities. The public key of the election is derived from the private key.
2. Voters encrypt their vote using the public key and post them in the bulletin board.
3. **Encrypted votes are multiplied (by candidate).**
4. The election authorities join their shares and recover the secret key to decrypt **only one ciphertext per candidate** and compute the tally.

What happens if the election authorities decrypt individual ballots instead of the ones resulting from multiplying them?

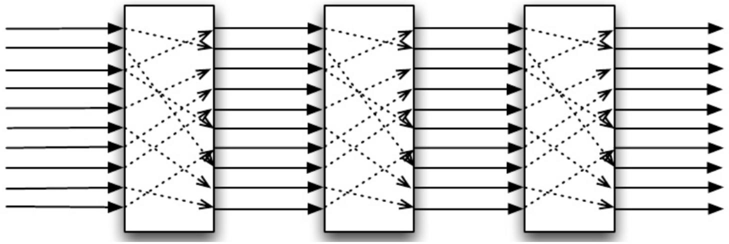
1. The election secret key is shared among a group of election authorities. The public key of the election is derived from the private key.
2. Voters encrypt their vote using the public key and post them in the bulletin board.
3. **Encrypted votes are multiplied (by candidate).**
4. The election authorities join their shares and recover the secret key to decrypt **only one ciphertext per candidate** and compute the tally.

What happens if the election authorities decrypt individual ballots instead of the ones resulting from multiplying them?

The order in which data is stored in the bulletin board can be used to link the identity of the voter to the value of the vote, if the order in which voters cast their ballots is also observed.

Mixing

What is a mixnet?



Mixing networks (aka mixnets) are a tool that enables a collection of servers to take as input a collection of ciphertexts and to output the corresponding ciphertexts, re-encrypted and shuffled according to a secret permutation.

Re-randomized encryption

Without knowing the secret key, modify the randomness used in the encryption so that

- The plaintext stays the same
- The new encryption cannot be linked to the old one

Re-randomized encryption

Without knowing the secret key, modify the randomness used in the encryption so that

- The plaintext stays the same
- The new encryption cannot be linked to the old one

With (exponential) ElGamal:

Let $(c, d) = (g^r, g^m \cdot y^r)$ be an encryption of the plaintext g^m using randomness r.

Re-randomized encryption

Without knowing the secret key, modify the randomness used in the encryption so that

- The plaintext stays the same
- The new encryption cannot be linked to the old one

With (exponential) ElGamal:

Let $(c, d) = (g^r, g^m \cdot y^r)$ be an encryption of the plaintext g^m using randomness r.
We take an encryption (under the same public key y) of 1 using a random value r',

$$(c_1, d_1) = (g^{r'}, g \cdot y^{r'}).$$

Re-randomized encryption

Without knowing the secret key, modify the randomness used in the encryption so that

- The plaintext stays the same
- The new encryption cannot be linked to the old one

With (exponential) ElGamal:

Let $(c, d) = (g^r, g^m \cdot y^r)$ be an encryption of the plaintext g^m using randomness r.
We take an encryption (under the same public key y) of 1 using a random value r',

$$(c_1, d_1) = (g^{r'}, g \cdot y^{r'}).$$

By multiplying these two ciphertexts

$$(c, d) \cdot (c_1, d_1) = (g^{r+r'}, g^m \cdot y^{r+r'})$$

we obtain an encryption of the same plaintext g^m but using a different randomness.

1. The election secret key is shared among a group of election authorities. The public key of the election is derived from the private key.
2. Voters encrypt their vote using the public key and post them in the bulletin board.
3. **At the end of the election, the mixnet takes those ciphertexts and re-encrypts and permutes them.**
4. The resulting encrypted votes are multiplied (by candidate).
5. The election authorities join their shares and recover the secret key to decrypt only one ciphertext per candidate and compute the tally.

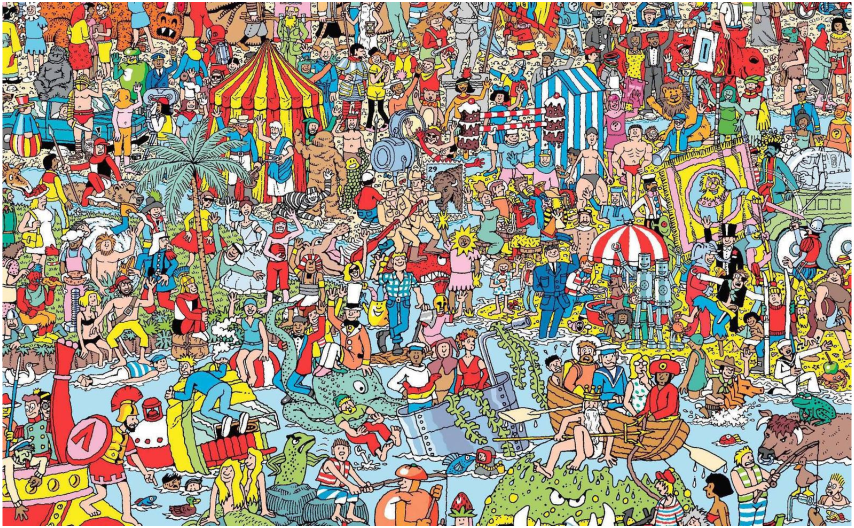
1. The election secret key is shared among a group of election authorities. The public key of the election is derived from the private key.
2. Voters encrypt their vote using the public key and post them in the bulletin board.
3. **At the end of the election, the mixnet takes those ciphertexts and re-encrypts and permutes them.**
4. The resulting encrypted votes are multiplied (by candidate).
5. The election authorities join their shares and recover the secret key to decrypt only one ciphertext per candidate and compute the tally.

How can we make sure that everybody (voters, mixnet nodes, election authorities, etc) is following the protocol and not cheating?

Zero-knowledge proof of knowledge

Where is Waldo?

How can I prove to you that I know where Waldo is, without telling you where?



Proofs

In a proof, a Prover wants to convince someone else (a Verifier) about something.

If I know that X is true, and I want to convince you of that, I'll try to present all the facts I know and the inferences from that fact that imply that X is true.

Proofs

In a proof, a Prover wants to convince someone else (a Verifier) about something.

If I know that X is true, and I want to convince you of that, I'll try to present all the facts I know and the inferences from that fact that imply that X is true.

Example: How can I prove that a number is not prime?

Proofs

In a proof, a Prover wants to convince someone else (a Verifier) about something.

If I know that X is true, and I want to convince you of that, I'll try to present all the facts I know and the inferences from that fact that imply that X is true.

Example: How can I prove that a number is not prime?

To prove that I know that 38477 is not prime, I will give you its factors, 109 and 353, and show you that indeed $38477 = 109 \cdot 353$.

Zero-knowledge proofs

Typically, a proof yields **some knowledge**, beyond the fact that the statement is true.

In the example, we learned not only that 38477 is not a prime, but we also learned its factorization.
Zero-knowledge proofs try to avoid this.

Zero-knowledge proofs

Typically, a proof yields **some knowledge**, beyond the fact that the statement is true.

In the example, we learned not only that 38477 is not a prime, but we also learned its factorization.
Zero-knowledge proofs try to avoid this.

Idea: Alice will prove to Bob that a statement X is true, Bob will be convinced of that, but he will not learn anything as a result of this process.

Zero-knowledge proofs

Typically, a proof yields **some knowledge**, beyond the fact that the statement is true.

In the example, we learned not only that 38477 is not a prime, but we also learned its factorization.
Zero-knowledge proofs try to avoid this.

Idea: Alice will prove to Bob that a statement X is true, Bob will be convinced of that, but he will not learn anything as a result of this process.

Properties we expect from ZK-POK:

- **Completeness**: if the statement is true, the verifier should always accept.
- **Soundness**: if the statement is false, the verifier should reject with a high probability.
- **Zero-knowledge**: the verifier should not learn anything beyond the validity of the statement. We say the verifier has *gained knowledge* from the interaction if he can easily compute something that he couldn't efficiently compute before the interaction.

Schnorr's proof (I)

Schnorr suggested the following interactive zero-knowledge proof of knowledge for the discrete logarithm.

Public parameters: a group G of order q, with generator g, and an element $h \in G$.

Statement the prover wants to prove: I know the discrete logarithm of h with respect to g.

Schnorr's proof (I)

Schnorr suggested the following interactive zero-knowledge proof of knowledge for the discrete logarithm.

Public parameters: a group G of order q, with generator g, and an element $h \in G$.

Statement the prover wants to prove: I know the discrete logarithm of h with respect to g.

Verifier	**Prover** $x : g^x = h$

Schnorr's proof (I)

Schnorr suggested the following interactive zero-knowledge proof of knowledge for the discrete logarithm.

Public parameters: a group G of order q, with generator g, and an element $h \in G$.

Statement the prover wants to prove: I know the discrete logarithm of h with respect to g.

Verifier		**Prover** $x : g^x = h$	
	$\xleftarrow{a=g^r}$	$r \overset{R}{\leftarrow} \mathbb{Z}_q$	(commitment)

Schnorr's proof (I)

Schnorr suggested the following interactive zero-knowledge proof of knowledge for the discrete logarithm.

Public parameters: a group G of order q, with generator g, and an element $h \in G$.

Statement the prover wants to prove: I know the discrete logarithm of h with respect to g.

$$\begin{array}{lccccl}
 & \textbf{Verifier} & & \textbf{Prover } x : g^x = h & \\
 & & \xleftarrow{a=g^r} & r \xleftarrow{R} \mathbb{Z}_q & \text{(commitment)} \\
\text{(challenge)} & b \xleftarrow{R} \mathbb{Z}_q & \xrightarrow{b} & &
\end{array}$$

Schnorr's proof (I)

Schnorr suggested the following interactive zero-knowledge proof of knowledge for the discrete logarithm.

Public parameters: a group G of order q, with generator g, and an element $h \in G$.

Statement the prover wants to prove: I know the discrete logarithm of h with respect to g.

	Verifier		**Prover** $x : g^x = h$	
		$\xleftarrow{a=g^r}$	$r \xleftarrow{R} \mathbb{Z}_q$	(commitment)
(challenge)	$b \xleftarrow{R} \mathbb{Z}_q$	$\xrightarrow{b}$		
		$\xleftarrow{c}$	$c = r + xb \pmod{q}$	(response)

Schnorr's proof (I)

Schnorr suggested the following interactive zero-knowledge proof of knowledge for the discrete logarithm.

Public parameters: a group G of order q, with generator g, and an element $h \in G$.

Statement the prover wants to prove: I know the discrete logarithm of h with respect to g.

$$\begin{array}{lcccl}
 & \textbf{Verifier} & & \textbf{Prover } x : g^x = h & \\
 & & \xleftarrow{a=g^r} & r \overset{R}{\longleftarrow} \mathbb{Z}_q & \text{(commitment)} \\
\text{(challenge)} & b \overset{R}{\longleftarrow} \mathbb{Z}_q & \xrightarrow{b} & & \\
 & & \xleftarrow{c} & c = r + xb \pmod{q} & \text{(response)} \\
 & \begin{array}{c}\text{accept iff:}\\ ah^b = g^c\end{array} & & &
\end{array}$$

Schnorr's proof (II)

Properties of Schnorr's proof:

- Completeness: if $g^x = h$, then the Verifier will be always convinced.

Schnorr's proof (II)

Properties of Schnorr's proof:

- Completeness: if $g^x = h$, then the Verifier will be always convinced.
- Soundness: if the Prover doesn't know such x, the Verifier will reject with high probability.

Schnorr's proof (II)

Properties of Schnorr's proof:

- Completeness: if $g^x = h$, then the Verifier will be always convinced.
- Soundness: if the Prover doesn't know such x, the Verifier will reject with high probability.

- What did the Verifier learn from the proof?

Schnorr's proof (II)

Properties of Schnorr's proof:

- Completeness: if $g^x = h$, then the Verifier will be always convinced.
- Soundness: if the Prover doesn't know such x, the Verifier will reject with high probability.
- What did the Verifier learn from the proof?
  He only learns that the Prover knows such x.

Schnorr's proof (II)

Properties of Schnorr's proof:

- Completeness: if $g^x = h$, then the Verifier will be always convinced.
- Soundness: if the Prover doesn't know such x, the Verifier will reject with high probability.
- What did the Verifier learn from the proof?
  He only learns that the Prover knows such x.

Schnorr proof is called a **Sigma protocol**. Sigma protocols have some interesting properties:

- can be repeated in parallel
- can be nicely combined to prove *I know a witness for x OR/AND for x'*.
- can be transformed into non-interactive zero-knowledge proofs.

Sigma Protocols in e-voting

POK from the voter:

- Proves: that the encrypted vote indeed contains one of the valid values (i.e. 0 or 1), without revealing which of them.
- It doesn't reveal: the value itself.

Sigma Protocols in e-voting

POK from the voter:

- Proves: that the encrypted vote indeed contains one of the valid values (i.e. 0 or 1), without revealing which of them.
- It doesn't reveal: the value itself.

POK by the election authorities:

- Proves: that the decryption is correct, that is:
  - they used the correct private key corresponding to the election public key,
  - the value they claim to be the result of the election indeed corresponds to the counting of votes present in the bulletin board.
- It doesn't reveal: the election private key.

Sigma Protocols in e-voting

POK from the voter:

- Proves: that the encrypted vote indeed contains one of the valid values (i.e. 0 or 1), without revealing which of them.
- It doesn't reveal: the value itself.

POK by the election authorities:

- Proves: that the decryption is correct, that is:
  - they used the correct private key corresponding to the election public key,
  - the value they claim to be the result of the election indeed corresponds to the counting of votes present in the bulletin board.
- It doesn't reveal: the election private key.

POK by each mixnet node:

- Proves: that the re-encryption and shuffling have been done correctly.
- It doesn't reveal: the randomness used nor the permutation applied to the ciphertexts.

Some e-voting systems

Helios voting sytem

Helios voting system

- Introduced in 2008 by Ben Adida. It is one of the e-voting systems more studied by academics.
- Web application for Internet voting: https://vote.heliosvoting.org
- It is easy to use, open-source, provides end-to-end verifiability.
- It constitutes a tool to support elections for companies, online groups, etc. It is customizable (authentication, look-and-feel, translations).

Election process (I)

System initialization

1. The user creates the election by setting the parameters and the list of eligible voters.
2. The software generates the ballot, private key, and public key.

Election process (II)

Vote casting

1. Every voter receives an e-mail containing her username, password, and the URL of the election.
2. The Javascript application starts and downloads the public election parameters.
3. The voter fills out the ballot, which is then encrypted by the application.
4. A hash of the encrypted vote is shown to the voter (receipt).
5. The voter has the option to audit the ballot. In this case the audited ballot cannot be cast, and the voter should start the vote casting process again.
6. The voter authenticates herself into the election system.
7. The voter ID, password, the encrypted vote and the corresponding ZK-POK are sent to the server.

Election process (III)

Tally and publication of votes

1. The Helios server publishes the encrypted votes, hashes and corresponding ZK-POK on the bulletin board (website).
2. The server computes the election outcome with homomorphic tally (no mixing).

EVA

Internet voting in Norway

Norway used Internet voting in the local elections of 2011 and in the parliamentary elections of 2013.

Internet voting in Norway

Norway used Internet voting in the local elections of 2011 and in the parliamentary elections of 2013.

Voter registration

Voters had to register their mobile phones with a centralized government register.

Voters receive a special card, delivered through the postal service, with personalized numeric **return codes**.

Internet voting in Norway

Norway used Internet voting in the local elections of 2011 and in the parliamentary elections of 2013.

Voter registration

Voters had to register their mobile phones with a centralized government register.

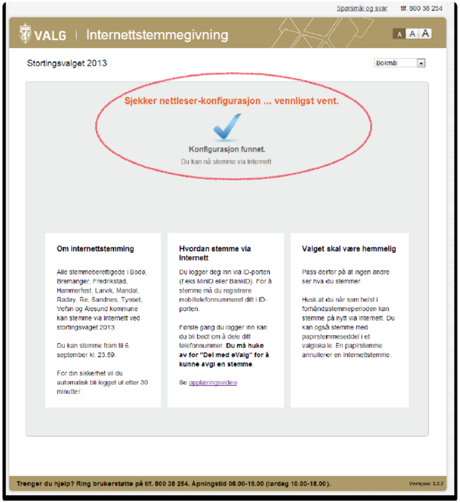
Voters receive a special card, delivered through the postal service, with personalized numeric **return codes**.

Return codes (cast-as-intended verification)

Return codes are four-digit numbers corresponding to each party running for election, randomly assigned for every voter.

Voting phase (I)

1. When ready to vote, the voter accesses a Javascript-based voting website from his/her browser.


Spørsmål og svar tlf. 800 38 254

VALG | Internettstemmegivning

Stortingsvalget 2013

Bokmål

Sjekker nettleser-konfigurasjon ... vennligst vent.

Konfigurasjon funnet.

Du kan nå stemme via Internett.

Om internettstemming

Alle stemmeberettigede i Bodø, Bremanger, Fredrikstad, Hammerfest, Larvik, Mandal, Radøy, Re, Sandnes, Tynset, Vefsn og Ålesund kommune kan stemme via Internett ved stortingsvaget 2013.

Du kan stemme fram til 6. september kl. 23.59.

For din sikkerhet vil du automatisk bli logget ut etter 30 minutter.

Hvordan stemme via Internett

Du logger deg inn via ID-porten (f.eks MinID eller BankID). For å stemme må du registrere mobiltelefonnummeret ditt i ID-porten.

Første gang du logger inn kan du bli bedt om å dele ditt telefonnummer. **Du må huke av for "Del med eValg" for å kunne avgi en stemme.**

Se opplæringsvideo.

Valget skal være hemmelig

Pass derfor på at ingen andre ser hva du stemmer.

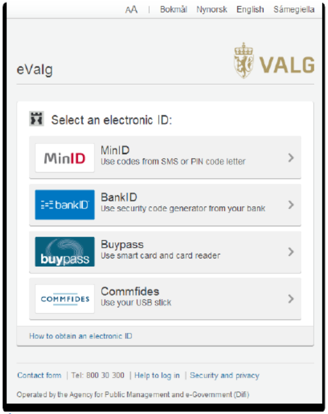
Husk at du når som helst i forhåndsstemmeperioden kan stemme på nytt via Internett. Du kan også stemme med papirstemmeseddel i et valglokale. En papirstemme annullerer en internettstemme.

Trenger du hjelp? Ring brukerstøtte på tlf. 800 38 254. Åpningstid 08.00-19.00 (lørdag 10.00-18.00).

Versjon: 3.2.7

Voting phase (II)

2. The voter is presented with the option of using one of several existing authentication services to confirm their identity (banking, smartcard, or the government MinID issued service).



AA | Bokmål Nynorsk English Sámegiella

eValg

VALG

Select an electronic ID:

MinID
Use codes from SMS or PIN code letter

BankID
Use security code generator from your bank

Buypass
Use smart card and card reader

Commfides
Use your USB stick

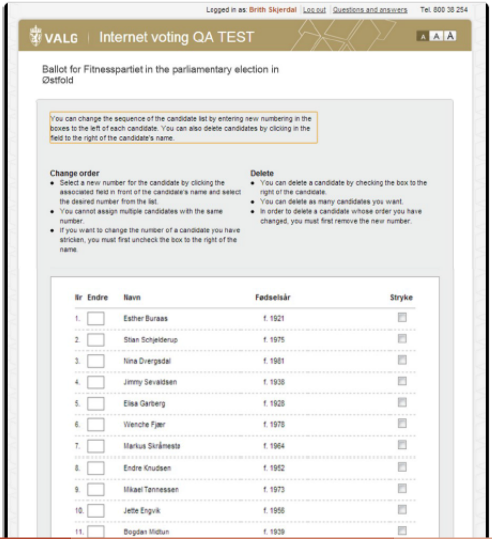
How to obtain an electronic ID

Contact form | Tel: 800 30 300 | Help to log in | Security and privacy

Operated by the Agency for Public Management and e-Government (Difi)

Voting phase (III)

3. The voter selects his/her choice and submits the ballot. The choice is accepted in the Vote Collection Server (VCS).



Logged in as: Brith Skjerdal | Log out | Questions and answers | Tel. 800 38 254

VALG | Internet voting QA TEST

Ballot for Fitnesspartiet in the parliamentary election in Østfold

You can change the sequence of the candidate list by entering new numbering in the boxes to the left of each candidate. You can also delete candidates by clicking in the field to the right of the candidate's name.

Change order
- Select a new number for the candidate by clicking the associated field in front of the candidate's name and select the desired number from the list.
- You cannot assign multiple candidates with the same number.
- If you want to change the number of a candidate you have stricken, you must first uncheck the box to the right of the name.

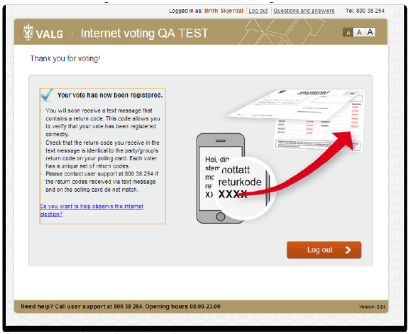
Delete
- You can delete a candidate by checking the box to the right of the candidate.
- You can delete as many candidates you want.
- In order to delete a candidate whose order you have changed, you must first remove the new number.

Nr	Endre	Navn	Fødselsår	Stryke
1.	☐	Esther Buraas	f. 1921	☐
2.	☐	Stian Schjelderup	f. 1975	☐
3.	☐	Nina Dvergsdal	f. 1981	☐
4.	☐	Jimmy Sevaldsen	f. 1938	☐
5.	☐	Elisa Garberg	f. 1928	☐
6.	☐	Wenche Fjær	f. 1978	☐
7.	☐	Markus Skråmestø	f. 1964	☐
8.	☐	Endre Knudsen	f. 1952	☐
9.	☐	Mikael Tønnessen	f. 1973	☐
10.	☐	Jette Engvik	f. 1956	☐
11.	☐	Bogdan Midtun	f. 1939	☐

Voting phase (IV)

4. The Vote Collection Server communicates with the Return Code Generator Server, which sends back an SMS text to the voter with the appropriate personalized return code.

   The voter then can match that code against his/her list of codes.


Logged in as: Brith Skjerdal | Log out | Questions and answers   Tel. 800 38 254

VALG | Internet voting QA TEST

Thank you for voting!

Your vote has now been registered.

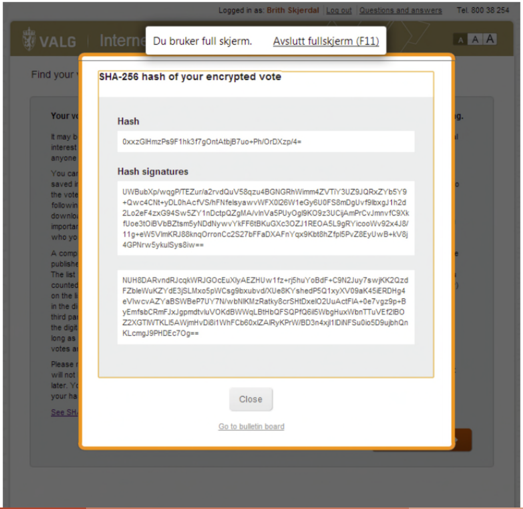
- You will soon receive a text message that contains a return code. This code allows you to verify that your vote has been registered correctly.
- Check that the return code you receive in the text message is identical to the party/group's return code on your polling card. Each voter has a unique set of return codes.
- Please contact user support at 800 38 254 if the return codes received via text message and on the polling card do not match.

Do you want to help observe the Internet election?

Need help? Call user support at 800 38 254. Opening hours 08.00-23.00

Voting phase (V)

5. The voter is presented with a SHA-256 of his/her encrypted vote and signature, that can be used to verify that the vote has been stored as cast in the GitHub repository (stored-as-cast).



Logged in as: Brith Skjerdal | Log out | Questions and answers   Tel. 800 38 254

Du bruker full skjerm.   Avslutt fullskjerm (F11)

SHA-256 hash of your encrypted vote

Hash

```
0xxzGlHmzPs9F1hk3f7gOntAtbjB7uo+Ph/OrDXzp/4=
```

Hash signatures

```
UWBubXp/wqgP/TEZur/a2rvdQuV58qzu4BGNGRhWimm4ZVTiY3UZ9JQRxZYb5Y9
+Qwc4CNt+yDL0hAcfVS/hFNfelsyawvWFX0l26W1eGy6U0FS8mDgUvf9lbxgJ1h2d
2Lo2eF4zxG94Sw5ZY1nDctpQZgMA/vlnVa5PUyOgl9KO9z3UCijAmPrCvJmnvfC9Xk
fUoe3tOiBVbBZtsm5yNDdNywvYkFF6tBKuGXc3OZJ1REOA5L9gRYicooWv92x4J8/
11g+eW5VlmKRJ88knqOrronCc2S27bFFaDXAFnYqx9Kbt8hZfpl5PvZ8EyUwB+kV8j
4GPNrw5ykulSys8iw==
```

```
NUH8DARvndRJcqkWRJGOcEuXlyAEZHUw1fz+rj5huYoBdF+C9N2Juy7swjKK2Qzd
FZbleWuKZYdE3jSLMxo5pWCsg9bxubvd/XUe8KYshedP5Q1xyXV09aK45iERDHg4
eVlwcvAZYaBSWBeP7UY7N/wbNlKMzRatky8crSHtDxelO2UuActFIA+0e7vgz9p+B
yEmfsbCRmFJxJgpmdtvluVOKdBWWqLBtHbQFSQPfQ6il5WbgHuxWbnTTuVEf2lBO
Z2XGTlWTKLI5AWjmHvDi8i1WhFCb60xlZAlRyKPrW/BD3n4xjl1lDiNFSu0io5D9ujbhQn
KLcmgJ9PHDEc7Og==
```

Final election phase

This phase includes the Decryption and Counting Ceremony.

1. *Cleansing*: identifies the ballots to be counted.

   Input: electoral roll, electronic ballot box.
   Output: the ballots to be counted, ZK-POK of correct cleansing.

2. *Mixing*: cryptographically anonymizes the ballots.

   Input: cleansed ballot box.
   Output: mixed ballot box, ZK-POK of correct shuffling.

3. *E-counting*: decrypts and computes the final count.

   Input: decryption key, mixed ballot box.
   Output: electronic vote count, ZK-POK of correct decryption.

Properties of the system

Supplements paper-based voting.

Coercion-resistant

- **repeat voting**: voters can cast multiple electronic votes, and cancel them by voting on paper.

Individual verifiability:

- *cast-as-intended*: return-codes
- *stored-as-cast*: hash of the encrypted vote

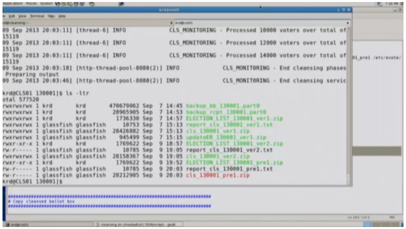
Universal verifiability:

- Zero-knowledge proofs proving, for example,
  - correct cleansing of ballots
  - correct mixing of ballots
  - correct decryption of ballots

Distribution of sensitive data

- Secret sharing: 6 out of 9 shares needed to reconstruct the private key.

The decryption and counting ceremony (I)

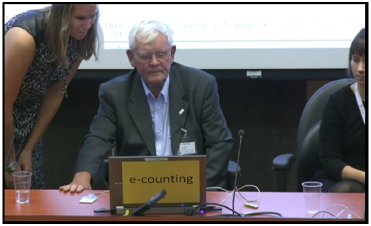


```
09 Sep 2013 20:03:11] [thread-6] INFO           CLS_MONITORING - Processed 10000 voters over total of
15119
09 Sep 2013 20:03:11] [thread-6] INFO           CLS_MONITORING - Processed 12000 voters over total of
15119
09 Sep 2013 20:03:11] [thread-6] INFO           CLS_MONITORING - Processed 14000 voters over total of
15119
09 Sep 2013 20:03:18] [http-thread-pool-8080(2)] INFO           CLS_MONITORING - End cleansing phases
 Preparing output
09 Sep 2013 20:03:46] [http-thread-pool-8080(2)] INFO           CLS_MONITORING - End cleansing servic

krd@CLS01 130001]$ ls -ltr
otal 577520
rwxrwxrwx 1 krd       krd       470679062 Sep  7 14:45 backup_bb_130001.part0
rwxrwxrwx 1 krd       krd        28965905 Sep  7 14:53 backup_rcpt_130001.part0
rwxrwxrwx 1 krd       krd         1736330 Sep  7 14:57 ELECTION_LIST_130001_ver1.zip
rwxrwxrwx 1 glassfish glassfish     10753 Sep  7 15:13 report_cls_130001_ver1.txt
rwxrwxrwx 1 glassfish glassfish  28426882 Sep  7 15:13 cls_130001_ver1.zip
rwxrwxrwx 1 glassfish glassfish    945499 Sep  7 15:15 updateER_130001_ver1.zip
rwxr-xr-x 1 krd       krd         1769622 Sep  9 18:57 ELECTION_LIST_130001_ver2.zip
rw-r----- 1 glassfish glassfish     10785 Sep  9 19:05 report_cls_130001_ver2.txt
rwxrwxrwx 1 glassfish glassfish  28158367 Sep  9 19:05 cls_130001_ver2.zip
rwxr-xr-x 1 krd       krd         1769622 Sep  9 19:52 ELECTION_LIST_130001_prel.zip
rw-r----- 1 glassfish glassfish     10785 Sep  9 20:03 report_cls_130001_prel.txt
rw-r----- 1 glassfish glassfish  28212905 Sep  9 20:03 cls_130001_prel.zip
krd@CLS01 130001]$
```

```
################################################################
# Copy cleansed ballot box
################################################################
```

The decryption and counting ceremony (II)



Conclusion

E-voting is a true reality in several countries.

There is some hope that a secure and practical voting system will exist some day... but there are still things to be improved.

Want to know more? See you at next week's event of Science and Cocktails: *Securing Digital Democracy*, in Christiania.