

Weaknesses in the Moscow Internet voting system

Pierrick Gaudry

CNRS, UNIVERSITÉ DE LORRAINE, INRIA
NANCY, FRANCE

Joint work with Alexander Golovnev (Harvard)

Special thanks to Julia Krivososova (TUT, Estonia)

ECC 2019, Bochum

Plan

Context

Attacks on the encryption scheme

Weaknesses in the general protocol

What would be the ideal e-voting system?

What kind of election?

September 8, 2019: day of local elections in Russia.

Moscow: election of the City Parliament (Moscow Duma).

In Moscow, there is also a City Government (headed by the Mayor of Moscow) for the executive part.

Rules:

- 45 members, each one elected by one district.
- Around 165,000 voters in each district; 7.3 M in total;
- In each district, the candidate with the largest number of votes gets the seat.

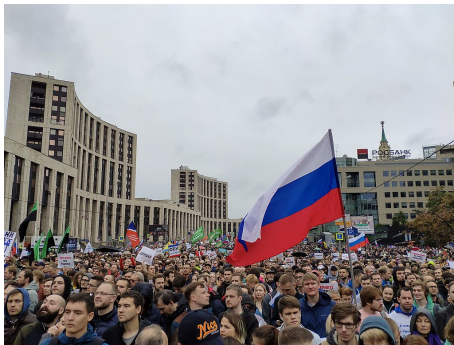
Internet voting experiment:

- Only in 3 districts;
- Any voter could register and use **Internet voting**;



Not exactly a peaceful context

Several **protests** in July / August due to rejection of opposition candidacies.



© Putnik / Wikipedia

Up to **20,000 participants** in Moscow.

Public test of the system

A **public testing** was organized, with a **bounty program** of up to 2 millions rubles (\approx \$30,000).

Source code made public on GitHub.

<https://github.com/moscow-technologies/blockchain-voting>

Various attack scenarios were proposed.

https://www.mos.ru/upload/.../5381/Formal_Offer.pdf

Public test of the system

A **public testing** was organized, with a **bounty program** of up to 2 millions rubles (\approx \$30,000).

Source code made public on GitHub.

<https://github.com/moscow-technologies/blockchain-voting>

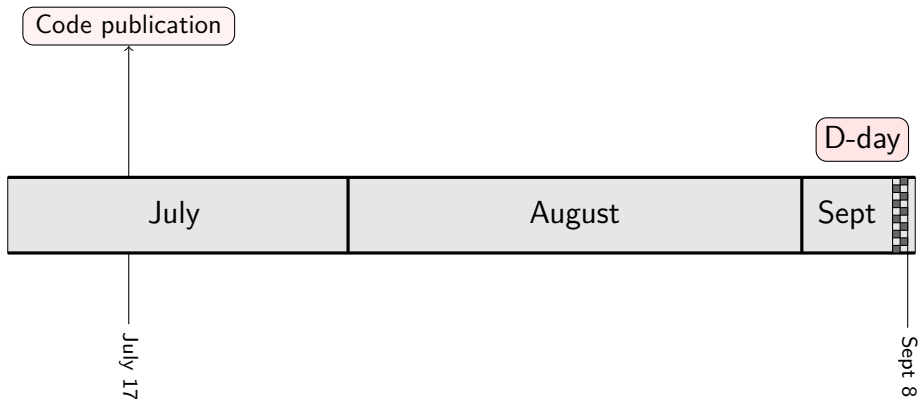
Various attack scenarios were proposed.

https://www.mos.ru/upload/.../5381/Formal_Offer.pdf

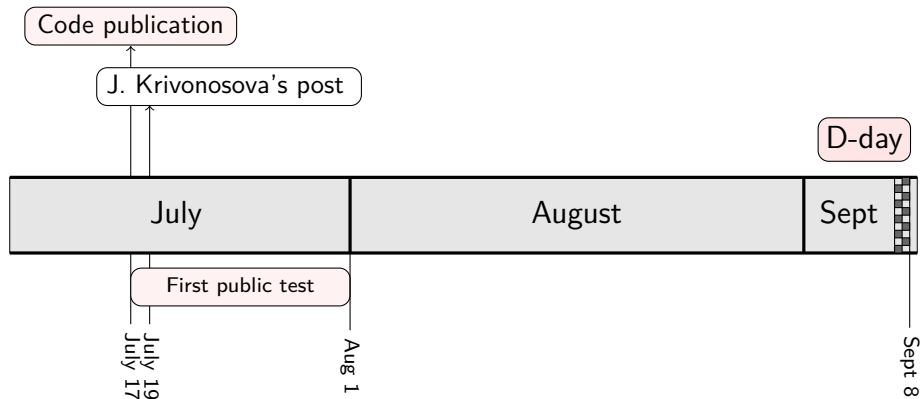
No specification! No documentation!

Rem. This is not part of a formal certification process.

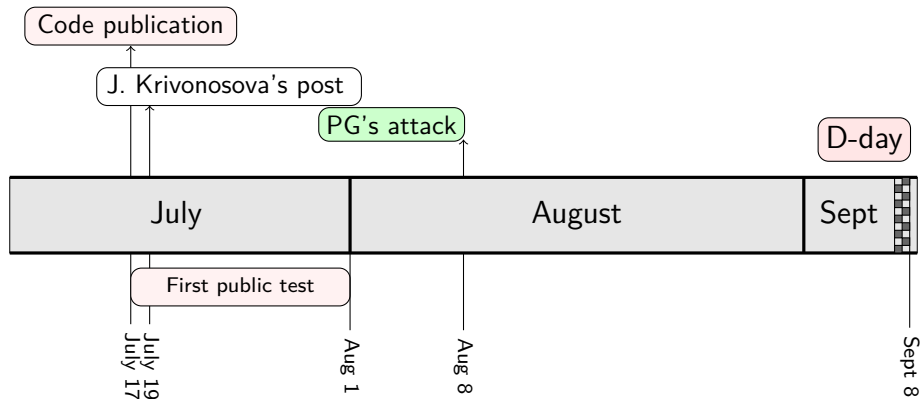
Timeline of Summer 2019



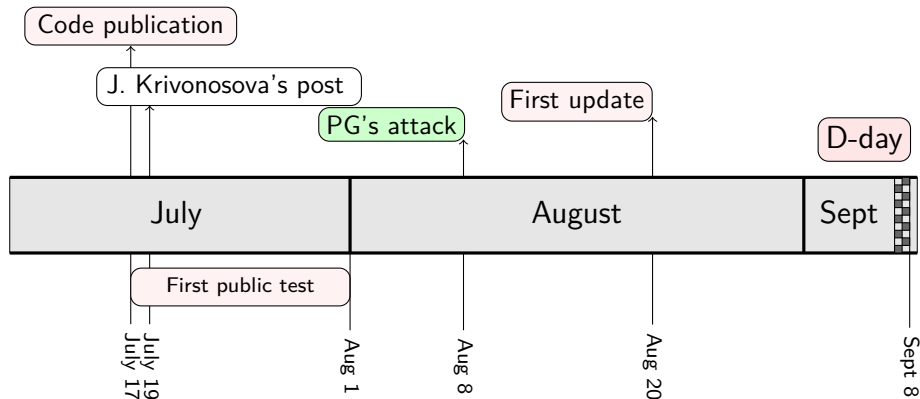
Timeline of Summer 2019



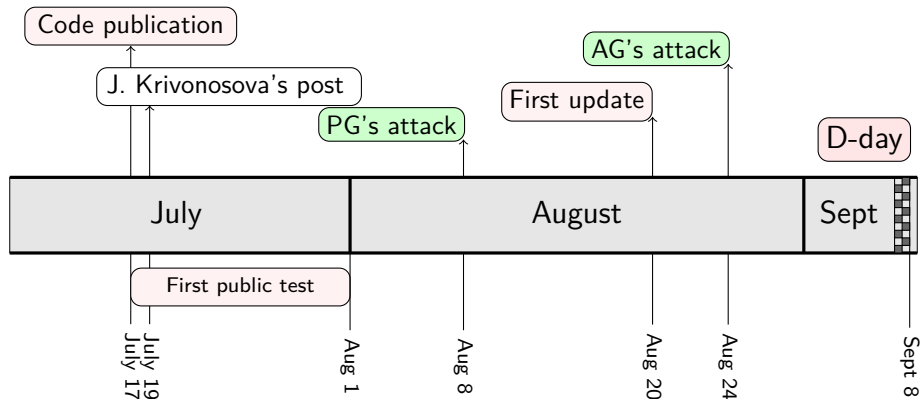
Timeline of Summer 2019



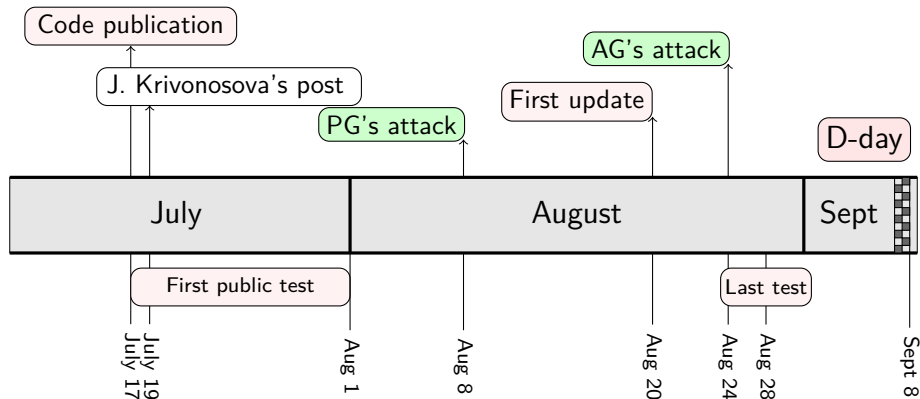
Timeline of Summer 2019



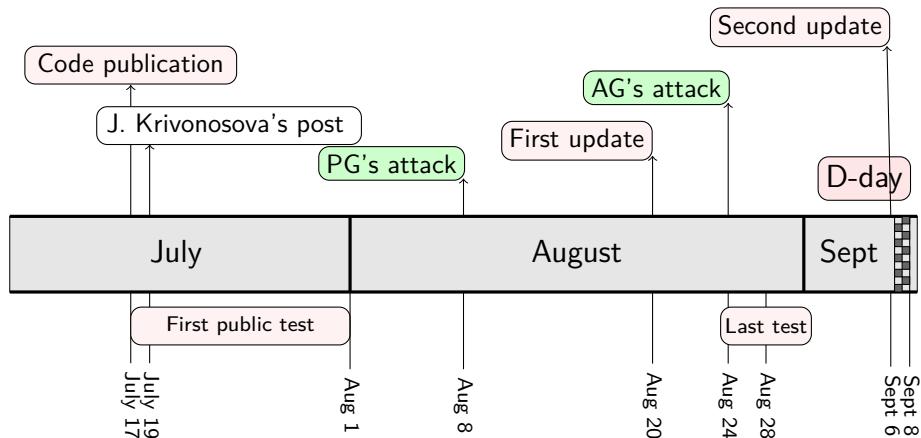
Timeline of Summer 2019



Timeline of Summer 2021



Timeline of Summer 2019



Plan

Context

Attacks on the encryption scheme

Weaknesses in the general protocol

What would be the ideal e-voting system?

Quick recall of ElGamal

Encryption scheme: variant of ElGamal in $\mathbb{Z}/p\mathbb{Z}$, with safe primes (Sophie Germain).

Let g be a generator, and $(\text{sk}, \text{pk} = g^{\text{sk}})$ a key-pair.

The **plain ElGamal encryption** of a message m is:

$$\text{Enc}_{g,\text{pk}}(m) = (a, b) = (g^r, \text{pk}^r \cdot m),$$

where r is a random (to be used only once).

The **decryption** using sk is:

$$\text{Dec}_{g,\text{sk}}(a, b) = b \cdot a^{-\text{sk}} = m.$$

If done correctly, this gives IND-CPA security.

A triple-ElGamal (encryption)

In the original scheme, there is a **multi-level** variant:

Choose $p_1 < p_2 < p_3$, three safe primes, together with 3 generators g_1, g_2, g_3 .

The **keys** are

$$\text{sk} = (\text{sk}_1, \text{sk}_2, \text{sk}_3); \quad \text{pk} = (g_1^{\text{sk}_1}, g_2^{\text{sk}_2}, g_3^{\text{sk}_3}),$$

The **encryption** of a message $m \in \mathbb{Z}/p_1\mathbb{Z}$ is obtained by

$$\begin{aligned} (a_1, b_1) &:= \text{Enc}_{g_1, \text{pk}_1}(m); & \text{map } a_1 \text{ to } \mathbb{Z}/p_2\mathbb{Z}; \\ (a_2, b_2) &:= \text{Enc}_{g_2, \text{pk}_2}(a_1); & \text{map } a_2 \text{ to } \mathbb{Z}/p_3\mathbb{Z}; \\ (a_3, b_3) &:= \text{Enc}_{g_3, \text{pk}_3}(a_2), \end{aligned}$$

and the encrypted message is

$$\text{MultiEnc}(m) = (b_1, b_2, a_3, b_3).$$

Rem. All mapping are obtained by canonical lifting to \mathbb{Z} .

A triple-ElGamal (decryption)

Knowing sk , the operations can be reversed to **decrypt** m from (b_1, b_2, a_3, b_3) :

$$\begin{aligned}a_2 &:= \text{Dec}_{g_3, sk_3}(a_3, b_3); & \text{map } a_2 \text{ to } \mathbb{Z}/p_2\mathbb{Z}; \\a_1 &:= \text{Dec}_{g_2, sk_2}(a_2, b_2); & \text{map } a_1 \text{ to } \mathbb{Z}/p_1\mathbb{Z}; \\m &:= \text{Dec}_{g_1, sk_1}(a_1, b_1).\end{aligned}$$

Due to the inequality $p_1 < p_2 < p_3$, **this works**.

A triple-ElGamal (decryption)

Knowing sk , the operations can be reversed to **decrypt** m from (b_1, b_2, a_3, b_3) :

$$\begin{aligned}a_2 &:= \text{Dec}_{g_3, sk_3}(a_3, b_3); & \text{map } a_2 \text{ to } \mathbb{Z}/p_2\mathbb{Z}; \\a_1 &:= \text{Dec}_{g_2, sk_2}(a_2, b_2); & \text{map } a_1 \text{ to } \mathbb{Z}/p_1\mathbb{Z}; \\m &:= \text{Dec}_{g_1, sk_1}(a_1, b_1).\end{aligned}$$

Due to the inequality $p_1 < p_2 < p_3$, **this works**.

Security.

Contrary to triple-DES where the security number of operations to break the system is squared, here it is just multiplied by 3.

Breaking the scheme is not harder than to
break the 3 underlying ElGamal independently.

A triple-ElGamal: why?

Why? We did not get the final answer.
But we speculated ...

All the p_i 's are chosen to be **less than 256 bits**.

This is **enforced** in the code with tests like

```
this.moduleP.compareTo(SOLIDITY_MAX_INT) >= 0
```

Solidity is the smart-contract language of **Ethereum**. And in the code, there is a decryption function written in this language.

The max size of native integers is indeed 256 bits.

Looks like the authors did not have the time / the competence to write a **multi-precision library** in Solidity and decided to “increase the security” in another way...

A triple-ElGamal: why?

Why? We did not get the final answer.
But we speculated ...

All the p_i 's are chosen to be **less than 256 bits**.

This is **enforced** in the code with tests like

```
this.moduleP.compareTo(SOLIDITY_MAX_INT) >= 0
```

Solidity is the smart-contract language. And in the
code, there is a decryption function in the language.

The max size of native integers is



Blockchain
inside!

Looks like the authors did not have the competence to
write a **multi-precision library** in Solidity and decided to “increase
the security” in another way...

First attack: solve DLP

Goal: compute a discrete logs mod a p of **256 bits**.

The challenge says it must be done in less than 12 hours.

Historically, first done publicly in ?? Let's check in **DLDB**.

First attack: solve DLP

Goal: compute a discrete logs mod a p of **256 bits**.

The challenge says it must be done in less than 12 hours.

Historically, first done publicly in ?? Let's check in **DLDB**.

1995-1996 (Weber, Denny, Zayer).

Current record: 795 bits (rump session yesterday!)

How much time does it take **today**? (must be less than 12 hours)

Computing discrete log in $\mathbb{Z}/p\mathbb{Z}$

- **SageMath 8.8** (uses GP/Pari internally for DLP).
Free software. Not bad for small sizes, but for 256 bits, did not finish after 4 days.
- **Magma 2.24-2**
Proprietary software, developed and sold by University of Sydney.
Faster than SageMath. But uses a lot of memory.
For 256 bits, 24 hours with 130 GB.
- **CADO-NFS** (rev. 6b3746a2e).
Free software, developed (mostly) in Nancy. Based on the Number Field Sieve algorithm.
For 256 bits, less than **10 minutes** in less than **1 GB**.

DLP with CADO-NFS

Running times on my 4-year old nothing-special desk PC:

key number	time
sk_1	425 sec
sk_2	507 sec
sk_3	314 sec

Each line includes 2 runs of CADO-NFS (one for g_i , one for pk_i); but many steps are (automatically) shared.

Rem. Variation from one key to another is not unusual for computations with moderately small primes.

Rem. This size was not thoroughly tested in CADO-NFS and sometimes it failed due to bad parameters in the descent phase. This was quickly fixed for the occasion.

First fix

The **first update** on August 20 did:

- **Remove** the triple-ElGamal encryption;
- **Upgrade** the key-size to 1024 bits;
- **Changed** the protocol, so that decryption is no longer part of a smart-contract (oh, oh!).
- Take generators in the **prime-order subgroup**.

The last item is the response to a side-remark that was made in the document of the first attack.

Second attack (by Golovnev)

Golovnev's attack:

- The generator is now in a prime-order subgroup. **Good!**
- But the messages are not. **Doh!**
- **One bit is leaked**: from the ciphertext, one knows if the message is a quadratic residue.
- From the source code: encrypted message = identifier of a candidate (no random nonce).

Quite often in e-voting, **one bit is the whole answer!**

Typical scenario:

- Two main candidates: pro-Putin vs opposition;
- One has an id that is a QR, not the other.

Second fix

Getting close to the election, the situation got **chaotic** (at least from my non-Russian speaking point of view):

- The developers seemed to **deny** that this second attack was real;
- Still, they **silently changed** the code, without updating GitHub;
- A **final public test** was done, and from the (minified) Javascript sent to the voters, they took care of the subgroup attack;
- Only 2 days before the election, GitHub was **updated**.

Plan

Context

Attacks on the encryption scheme

Weaknesses in the general protocol

What would be the ideal e-voting system?

Encryption fixed. Problem solved?

Weak encryption is one of the easiest thing to analyze from source code only, **without spec**.

For **other properties, speculations** based on discussions, press articles, source code (incl. minified JS).

General impression: the whole protocol is messed up!

- Privacy? X
- Verifiability? very partial
- Coercion resistance? X
- Vote buying? X

Thanks to us, they now have a **reinforced door** on a **house** made of **cardboard**.

Overview of the protocol

1. **Registration / authentication.**

Based on existing infrastructure for administration in Russia.
Not specific to this election.

2. **Voting.**

The voter connects to the server, gets the javascript, the choice is encrypted locally with the key of the election and sent back.

3. **Ballot box.**

The server sends the encrypted ballots to the blockchain, with no reference to the voter, in random order.

4. **End of the election.**

The trustees who own the decryption key (they used Shamir secret sharing) decrypt the ballots and put them in the blockchain as well.

Overview of the protocol

1. **Registration / authentication.**

Based on existing infrastructure for administration in Russia.
Not specific to this election.

2. **Voting.**

The voter connects to the server, gets the javascript, the choice is encrypted locally with the key of the election and sent back.

3. **Ballot box.**

The server sends the encrypted ballots to the blockchain, with no reference to the voter, in random order.

4. **End of the election.**

The trustees who own the decryption key (they used Shamir secret sharing) decrypt the ballots and put them in the blockchain as well.

They also publish the **decryption key**. (what ?!?!?!?)

Role of the encryption

In fact, for the designers:

- Privacy is “guaranteed” by the **server** cutting the link between ballot and voter;
- Encryption is here only to protect against revealing the **partial tally** during the election day.

Further remarks

On the **blockchain**:

- **No idea** why they wanted to have the decryption made in a **smart-contract**.
- The blockchain is a **private** one. It provides no guarantee whatsoever.
- During the election, the voters could query the blockchain via a web server and check that their ballot was there.

Interaction with the Moscow Department of Information Technology was **frustrating**:

- They answered only to some of my technical questions; only those that were not embarrassing.
- I could not get any documentation or specification of the protocol.

D-day

They did **not cancel** the use of Internet voting.

In total, about **10,000 votes** by Internet.

In one of the districts, the difference between the first and the second candidate was **less than 100 votes**.

A few hours after the election, the access to the blockchain was cut. (What ?!?!?!?)

Fortunately, journalists of the **Meduza online newspaper** (exiled in Riga) made a copy before the shutdown, with all the ballots and the decryption key.

One of the journalists who was a voter kept track of his network communication during the voting and **could indeed check** that his ballot had been taken into account.

Impact of our findings

Main impact of our findings is **not** fixing the encryption.

We publicly showed that this was **not the perfect scheme**, as claimed.

Large **press coverage** (thumbs up for Meduza!)

Moscow had to **acknowledge** the problems.

Hopefully made some voters **skeptical** and they decided to use **traditional voting**.

Hopefully, they get some **pressure** to do a **better system** for the next experiments.

Positive note: Opening (part of) the source code and doing public testing is to be **commended**.

Plan

Context

Attacks on the encryption scheme

Weaknesses in the general protocol

What would be the ideal e-voting system?

Too many required properties

The **ideal system** needs to provide (at least):

Privacy, end-to-end verifiability, coercion resistance, usability, accessibility, availability, accountability, . . .

And all of this with a well-defined and reasonable **trust model**.

This does not exist (yet?).

Any practical system currently needs to choose the best compromise for the given context.

Let's concentrate on **privacy**

Secret of the vote, privacy

Informal definition: my vote should be secret.

Difficulties to make this **formally** precise:

- The result is public (and leaks information about your vote);
- Sometimes, intermediate results are public (list of ballots; tally for each district; . . .)
- Trust assumptions (voting device; various “trustees”)

Main tool: threshold encryption.

Threshold encryption is not enough

How to cut the link voters \leftrightarrow ballot before decryption?

Crypto means:

- homomorphic encryption
- verifiable mixnets
- multi-party computation

Non-crypto means:

- servers should forget a lot of data (often in contradiction with redundancies / backups / ...)

Rem. Perfect forward secrecy vs PQ resistance.

Both care about an attack in the future, when crypto might get broken. No deployed system is PQ-resistant today.

How to be private wrt my computer?

Sometimes used as the final argument by anti-evoting zealots.

This is not possible!

How to be private wrt my computer?

Sometimes used as the final argument by anti-evoting zealots.

This is not possible!

Yes it is. At least in theory.

But will need an **additional channel or computing device**.

- In Switzerland, they receive some paper voting material by post (but they don't use it for this property);
- If the voter has two devices (one PC and one Smartphone), this opens many possibilities, especially with scanning QR-codes.

Security goal: none of your device knows your vote, unless they are both infected by pieces of malware that can communicate.
(See e.g. BeleniosVS, a not-implemented variant of Helios.)

Summary of privacy properties for 2 systems

	Enc	Cut link voter-vote	Cut link voter-ballot
Helios	n trust.	Hom.	–
Moscow	–	non-crypto	non-crypto

	Privacy wrt device	Privacy wrt server	Forward sec. wrt pub data
Helios	No	Yes/No	Yes/No
Moscow	No	No	Yes

Another very difficult property:

Coercion resistance

Coercion resistance

Internet voting is **remote voting**. Coercion is built-in!

Typical cases:

- **Threat:** “Vote like this or you’ll have problems, you know. . . .”
- **Vote buying:** “I sell my vote for \$50; who wants to buy it?”
- **Family voting:** “Hey wife and kids! Let’s vote! Gimme your credentials!”

Rem. In all postal voting systems that I know, the problem exists as well.

In some contexts (like, yes, Moscow Duma election), the risk is high. (I heard some plausible stories; but no direct testimony.)

Coercion resistance: how?

Main hypothesis: the voter will have a period of time where she can vote freely.

Sometimes also require a period of time where the voter is free before the election opens.

Main idea: allow the voter to lie.



E.g. in Civitas, the voters can create by themselves fake voting material. The corresponding ballots will be blindly removed in the tally, with a ZKP that everything was done in the appropriate way.

Current state-of-the-art: this works in theory, but hard to make it practical (see survey at E-Vote-Id 2019).

Personal conclusion: don't use internet voting (yet) in a context where coercion is a concern.

More properties

- **Verifiability.** I should be sure that my vote is taken into account, and that everything went smoothly.
- **Usability.** Not all votes have a Master in Crypto...
- **Accessibility.** In principle, e-voting could be more friendly to voters with disabilities than paper voting. Or not...
- **Availability.** Backups, redundancy, resistance to DoS.
- **Accountability.** In case of problem (a verification fails), being able to tell who cheated is often very important.

Combining everything?

Unfortunately, **not realistic**, with current state of the art.

Things that can help:

- Having an **electronic ID** card helps for eligibility (cf Estonia);
- If we can afford sending **paper** material by post, this can help for verifiability and privacy w.r.t voting device (cf Switzerland);
- If the voters have several **independent devices**, this can help for verifiability and privacy w.r.t voting device (cf BeleniosVS);
- A **blockchain** might be a convenient way to implement a public append-only board;
- If all the voters have a degree in crypto AND are willing to read the documentation and to follow the protocol, life is much simpler.

IACR recent elections with Helios

Helios has a lot of nice properties . . . but it assumes that voters are doing their job.

Questions:

- Who participated in the **IACR 2019 election**?

IACR recent elections with Helios

Helios has a lot of nice properties . . . but it assumes that voters are doing their job.

Questions:

- Who participated in the **IACR 2019 election**?
- Did you check that your ballot was present in the public board? When?

IACR recent elections with Helios

Helios has a lot of nice properties . . . but it assumes that voters are doing their job.

Questions:

- Who participated in the **IACR 2019 election**?
- Did you check that your ballot was present in the public board? When?
- Did you use the “Audit” feature to check if you were not cheated by your machine?

IACR recent elections with Helios

Helios has a lot of nice properties . . . but it assumes that voters are doing their job.

Questions:

- Who participated in the **IACR 2019 election**?
- Did you check that your ballot was present in the public board? When?
- Did you use the “Audit” feature to check if you were not cheated by your machine?
- If you audited, did you put a random vote for the audited ballot? (how?)