On e-voting and privacy

Jan Willemsen

UT, Cybernetica
What is e-voting??

- A citizen sits in front of his computer,
What is e-voting??

- A citizen sits in front of his computer,
- opens a voting application (e.g. a web browser),
What is e-voting??

- A citizen sits in front of his computer,
- opens a voting application (e.g. a web browser),
- clicks an appropriate name.
Simple, isn’t it?

- No, it’s not.
Simple, isn’t it?

- No, it’s not.
- Vote transmission over public media (Internet, phone line) is not secure.
Simple, isn’t it?

- No, it’s not.
- Vote transmission over public media (Internet, phone line) is not secure.
- Thus we need to encrypt the votes.
Is it now OK?

- No, it’s not.
Is it now OK?

- No, it’s not.
- Some how we need to find out the sum of all votes.
Is it now OK?

- No, it’s not.
- Some how we need to find out the sum of all votes.
- How on Earth should that be possible if the votes are encrypted?
Should a server decrypt?

- A voting server could possess a decryption key for every voter. But …
Should a server decrypt?

- A voting server could possess a decryption key for every voter. But …
- The Estonian Riigikogu Valimise seadus §1 says:
  (2) Riigikogu liikmete valimised on vabad, üldised, ühetaolised ja otsesed. Hääletamine on salajane.
Should a server decrypt?

- A voting server could possess a decryption key for every voter. But …
- The Estonian Riigikogu Valimise seadus §1 says: (2) Riigikogu liikmete valimised on vabad, üldised, ühetaolised ja otsesed. Hääletamine on salajane.
- Can we claim privacy if some server can decode everything?
Should a server decrypt?

- A voting server could possess a decryption key for every voter. But …
- The Estonian Riigikogu Valimise seadus §1 says:
  (2) Riigikogu liikmete valimised on vabad, üldised, ühetaolised ja otsesed. Hääletamine on salajane.
- Can we claim privacy if some server can decode everything?
- Even threshold trust does not solve the essential problem – if $t + 1$ servers are compromised, the votes become public.
Homomorphic cryptography

- It is possible first to combine all the cryptograms of the votes to one large cryptogram and decode that one to obtain the sum of all of them.

No, as every single vote can be decoded just like the whole sum.
Homomorphic cryptography

- It is possible first to combine all the cryptograms of the votes to one large cryptogram and decode that one to obtain the sum of all of them.
- We need a special (so-called *homomorphic*) underlying cryptosystem for that (ElGamal, Paillier, Damgård-Jurik are fine)
Homomorphic cryptography

• It is possible first to combine all the cryptograms of the votes to one large cryptogram and decode that one to obtain the sum of all of them.

• We need a special (so-called *homomorphic*) underlying cryptosystem for that (ElGamal, Paillier, Damgård-Jurik are fine)

• Do they help?
Homomorphic cryptography

- It is possible first to combine all the cryptograms of the votes to one large cryptogram and decode that one to obtain the sum of all of them.
- We need a special (so-called homomorphic) underlying cryptosystem for that (ElGamal, Paillier, Damgård-Jurik are fine)
- Do they help?
- No, as every single vote can be decoded just like the whole sum.
Anything else …

- … doesn’t work either.
Anything else . . .

- . . . doesn’t work either.
- **Theorem.** If an electronic voting system is capable of decoding the result of voting by any subset of voters, it is possible to decode every single vote.
Anything else . . .

- . . . doesn’t work either.
- **Theorem.** If an electronic voting system is capable of decoding the result of voting by any subset of voters, it is possible to decode every single vote.
- **Proof.** Say, the set of voters is $X$. Take any $x \in X$ and decode $X$ together with $X \setminus \{x\}$. The difference of the results gives $x$’s vote.
Now what?

- The only way to try design a privacy-preserving voting system is to design it for a predetermined set of voters (so-called “boardroom voting”).
Now what?

- The only way to try design a privacy-preserving voting system is to design it for a predetermined set of voters (so-called “boardroom voting”).
- The good side: we do not have to be very concerned about the possibility that some party leaves the boardroom in the middle of the action.
Now what?

- The only way to try design a privacy-preserving voting system is to design it for a predetermined set of voters (so-called “boardroom voting”).
- The good side: we do not have to be very concerned about the possibility that some party leaves the boardroom in the middle of the action.
- The bad side: the resulting scheme is probably not very practical ...
Now what?

• The only way to try design a privacy-preserving voting system is to design it for a predetermined set of voters (so-called “boardroom voting”).

• The good side: we do not have to be very concerned about the possibility that some party leaves the boardroom in the middle of the action.

• The bad side: the resulting scheme is probably not very practical . . .

• . . . but still hopefully applicable in some limited setting.
Planning the protocol

• The voters should still encrypt their votes.
Planning the protocol

- The voters should still encrypt their votes.
- No-one else should possess the respective decryption keys.
Planning the protocol

- The voters should still encrypt their votes.
- No-one else should possess the respective decryption keys.
- Thus, the voters should decrypt their own votes.
 Planning the protocol

- The voters should still encrypt their votes.
- No-one else should possess the respective decryption keys.
- Thus, the voters should decrypt their own votes.
- Consequently, our protocol should contain (at least) two rounds.
Setting the protocol up

- Let us have the voters $A_1, A_2, \ldots, A_n$. 
Setting the protocol up

- Let us have the voters $A_1, A_2, \ldots, A_n$.
- Choose a group $G$ and an element $g$ of large order so that the respective discrete logarithm problem is hard.
Setting the protocol up

- Let us have the voters $A_1, A_2, \ldots, A_n$.
- Choose a group $G$ and an element $g$ of large order so that the respective discrete logarithm problem is hard.
- $\mathbb{Z}_p^*$ and its generator $g$ for a good choice of prime $p$ will do.
Setting the protocol up

- Let us have the voters $A_1, A_2, \ldots, A_n$.
- Choose a group $G$ and an element $g$ of large order so that the respective discrete logarithm problem is hard.
- $\mathbb{Z}_p^*$ and its generator $g$ for a good choice of prime $p$ will do.
- Each party $A_i$ chooses his vote $v_i$ and a random exponent invertible in $\mathbb{Z}_{p-1}$.
Protocol: encryption

- $A_1 : g^{a_1}$
Protocol: encryption

- $A_1 : g^{a_1}$
- $A_2 : (g^{a_1})^{a_2} = g^{a_1 a_2}$
Protocol: encryption

- $A_1 : g^{a_1}$
- $A_2 : (g^{a_1})^{a_2} = g^{a_1a_2}$
- ...

Protocol: encryption

- $A_1 : g^{a_1}$
- $A_2 : (g^{a_1})^{a_2} = g^{a_1 a_2}$
- ...
- $A_n : g^{a_1 a_2...a_n}$
Protocol: decryption

- \( A_1 : (g^{a_1 a_2 \ldots a_n}) a_1^{-1} v_1 = g^{v_1 a_2 \ldots a_n} \)

In order to obtain the result of the voting, we must solve "limited discrete logarithm problem" by raising \( g \) to all possible powers \( v_1 v_2 \ldots v_n \) and comparing the results to the output of the protocol.
Protocol: decryption

- $A_1 : (g^{a_1 a_2 \ldots a_n}) a_1^{-1} v_1 = g^{v_1 a_2 \ldots a_n}$
- $A_2 : (g^{v_1 a_2 \ldots a_n}) a_2^{-1} v_2 = g^{v_1 v_2 a_3 \ldots a_n}$
Protocol: decryption

- $A_1 : (g^{a_1 a_2 \ldots a_n})^{a_1^{-1} v_1} = g^{v_1 a_2 \ldots a_n}$
- $A_2 : (g^{v_1 a_2 \ldots a_n})^{a_2^{-1} v_2} = g^{v_1 v_2 a_3 \ldots a_n}$
- ...

In order to obtain the result of the voting, we must solve "limited discrete logarithm problem" by raising $g$ to all possible powers $v_1 v_2 \ldots v_n$ and comparing the results to the output of the protocol.
Protocol: decryption

- $A_1 : (g^{a_1 a_2 \ldots a_n})^{a_1^{-1} v_1} = g^{v_1 a_2 \ldots a_n}$
- $A_2 : (g^{v_1 a_2 \ldots a_n})^{a_2^{-1} v_2} = g^{v_1 v_2 a_3 \ldots a_n}$
- $\ldots$
- $A_n : g^{v_1 v_2 \ldots v_n}$

In order to obtain the result of the voting, we must solve "limited discrete logarithm problem" by raising $g$ to all possible powers $v_1 v_2 \ldots v_n$ and comparing the results to the output of the protocol.
Protocol: decryption

- $A_1 : (g^{a_1a_2\ldots a_n})a_1^{-1}v_1 = g^{v_1a_2\ldots a_n}$
- $A_2 : (g^{v_1a_2\ldots a_n})a_2^{-1}v_2 = g^{v_1v_2a_3\ldots a_n}$
- ...
- $A_n : g^{v_1v_2\ldots v_n}$

In order to obtain the result of the voting, we must solve “limited discrete logarithm problem” by raising $g$ to all possible powers $v_1v_2\ldots v_n$ and comparing the results to the output of the protocol.
All-against-one attack

- Say, $A_2, \ldots, A_n$ choose $a_2 = \ldots = a_n = 1$. 

Then $A_1$ computes $g^{a_1}$ in the first round and $(g^{a_1})^{a_1} = g^{v_1}$ in the second.

Then $v_1$ can be found by solving the limited discrete logarithm problem.

But hey, if $A_2, \ldots, A_n$ collaborate, they can find out $v_i$ anyway!

We have an interesting situation: in order for my vote to be secure, at least one other voter has to be honest!
All-against-one attack

- Say, $A_2, \ldots, A_n$ choose $a_2 = \ldots = a_n = 1$.
- Then $A_1$ computes $g^{a_1}$ in the first round and $(g^{a_1})^{a_1^{-1}v_1} = g^{v_1}$ in the second.
All-against-one attack

- Say, $A_2, \ldots, A_n$ choose $a_2 = \ldots = a_n = 1$.
- Then $A_1$ computes $g^{a_1}$ in the first round and $(g^{a_1})^{a_1^{-1}v_1} = g^{v_1}$ in the second.
- Then $v_1$ can be found by solving the limited discrete logarithm problem.
All-against-one attack

- Say, $A_2, \ldots, A_n$ choose $a_2 = \ldots = a_n = 1$.
- Then $A_1$ computes $g^{a_1}$ in the first round and $(g^{a_1})^{-1}v_1 = g^{v_1}$ in the second.
- Then $v_1$ can be found by solving the limited discrete logarithm problem.
- But hey, if $A_2, \ldots, A_n$ collaborate, they can find out $v_i$ anyway!
All-against-one attack

- Say, $A_2, \ldots, A_n$ choose $a_2 = \ldots = a_n = 1$.
- Then $A_1$ computes $g^{a_1}$ in the first round and $(g^{a_1})^{a_1^{-1}v_1} = g^{v_1}$ in the second.
- Then $v_1$ can be found by solving the limited discrete logarithm problem.
- But hey, if $A_2, \ldots, A_n$ collaborate, they can find out $v_i$ anyway!
- We have an interesting situation: *in order for my vote to be secure, at least one other voter has to be honest!*
Is one other honest guy enough?

- No, it’s not.
Is one other honest guy enough?

- No, it’s not.
- $A_n$ can give $g^{a_1}$ as his first round output as this value is public anyway.
Is one other honest guy enough?

- No, it’s not.
- \(A_n\) can give \(g^{a_1}\) as his first round output as this value is public anyway.
- In order to do it *legally*, \(A_n\) has to compute the true discrete logarithm

\[
\log_{g^{a_1}} g^{a_2 \ldots a_n}.
\]
Is one other honest guy enough?

- No, it’s not.
- $A_n$ can give $g^{a_1}$ as his first round output as this value is public anyway.
- In order to do it *legally*, $A_n$ has to compute the true discrete logarithm

$$\log g^{a_1} g^{a_2 \ldots a_n}.$$ 

- This can be avoided by requiring the proofs of knowledge of their own exponents from everybody.
Is one other honest guy enough?

- No, it’s not.
- $A_n$ can give $g^{a_1}$ as his first round output as this value is public anyway.
- In order to do it legally, $A_n$ has to compute the true discrete logarithm
  \[
  \log_{g^{a_1}} g^{a_2 \ldots a_n}.
  \]
- This can be avoided by requiring the proofs of knowledge of their own exponents from everybody.
- Zero-knowledge proofs can do the job.
Good and bad sides

+ The protocol is very efficient – only $2n$ modular exponents are needed to compute the result.
Good and bad sides

+ The protocol is very efficient – only $2n$ modular exponents are needed to compute the result
  - This is good compared to $2n^2 + 2n$ done in the protocol by Kiayias and Yung ...
Good and bad sides

+ The protocol is very efficient – only $2n$ modular exponents are needed to compute the result

  • This is good compared to $2n^2 + 2n$ done in the protocol by Kiayias and Yung ...
  • … and in a way as efficient as it can get – everybody has to perform at least 2 operations.
Good and bad sides

+ The protocol is very efficient – only $2n$ modular exponents are needed to compute the result
  - This is good compared to $2n^2 + 2n$ done in the protocol by Kiayias and Yung …
  - … and in a way as efficient as it can get – everybody has to perform at least 2 operations.

- The rounds have to be carried out in the predefined order, otherwise it may be possible to decode some votes.
Anything else wrong?

- Probably yes, at least points to be improved.
Anything else wrong?

- Probably yes, at least points to be improved.
- We could still try to cope with some parties failing to complete the protocol.
Anything else wrong?

- Probably yes, at least points to be improved.
- We could still try to cope with some parties failing to complete the protocol.
- $A_n$ learns the sum of other votes before the others do. He could change his mind before voting based on that information.

Etc. Security proofs/improvements are needed – open call for student contributions!
Anything else wrong?

- Probably yes, at least points to be improved.
- We could still try to cope with some parties failing to complete the protocol.
- $A_n$ learns the sum of other votes before the others do. He could change his mind before voting based on that information.
- Etc. Security proofs/improvements are needed – open call for student contributions!
That’s how far we are.

- Questions?