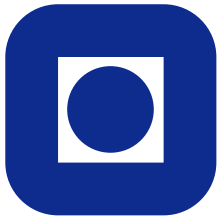


NTNU
Norwegian University of
Science and Technology

Electronic elections – National efforts

Kristian Gjøsteen

Dept. of Mathematical Sciences, NTNU

Kjeller – June 25, 2010

Contents

Analysing the cryptographic protocol to be used in Norwegian trials of electronic voting in 2011.

Proving Security

If it's provably secure, it's probably not. *– Lars Knudsen*

Provable security can be used to increase confidence in cryptographic protocols.

Proving Security

Provable security can be used to increase confidence in cryptographic protocols.

1. Proofs are usually relative to some complexity assumption and take the form of a reduction. If a «successful» attacker can be turned into an algorithm doing a computation we believe is impossible, we must believe no such attacker can exist.
    - The resources spent on studying integer factoring is significantly higher than the resources spent on any single protocol.
    - The security proof (reduction) must be correct.

Proving Security

Provable security can be used to increase confidence in cryptographic protocols.

2. We usually prove that a certain class of attackers cannot exist. This class is usually characterized by a security goal and a threat model (attacker capability). We need to get the class right.
    - Example I: There was a long debate about the «correct» class of attackers for public key encryption.

Proving Security

Provable security can be used to increase confidence in cryptographic protocols.

2. We usually prove that a certain class of attackers cannot exist. This class is usually characterized by a security goal and a threat model (attacker capability). We need to get the class right.
    - Example I: There was a long debate about the «correct» class of attackers for public key encryption.
    - Example II: The famous Needham-Schroeder protocol was proved secure, but against the wrong class of attackers.

Proving Security

Provable security can be used to increase confidence in cryptographic protocols.

2. We usually prove that a certain class of attackers cannot exist. This class is usually characterized by a security goal and a threat model (attacker capability). We need to get the class right.
    - Example I: There was a long debate about the «correct» class of attackers for public key encryption.
    - Example II: The famous Needham-Schroeder protocol was proved secure, but against the wrong class of attackers.
    - It may be impossible to protect against certain classes of attackers.

Proving Security

Provable security can be used to increase confidence in cryptographic protocols.

3. Some systems are provably insecure. Some systems are (conditionally) provably secure. Some systems are neither.
    - One may also search for «minimal» easy-to-analyse complexity assumptions necessary to prove a system secure.

The E-valg 2011 Project

Main goal:

> *Establish a secure electronic election system for parliamentary, municipal and county elections that increases accessibility for all voter groups. [...]*

The E-valg 2011 Project

Main goal:

> *Establish a secure electronic election system for parliamentary, municipal and county elections that increases accessibility for all voter groups. [...]*

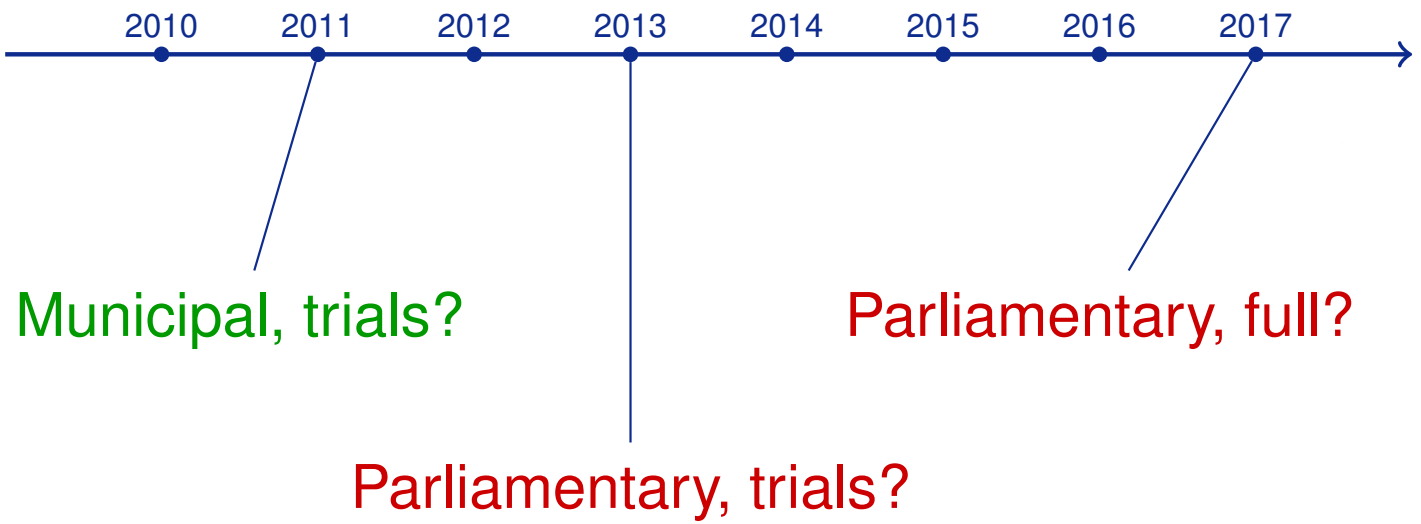
Note

There will only be advance internet voting. There will be no electronic voting on election day.

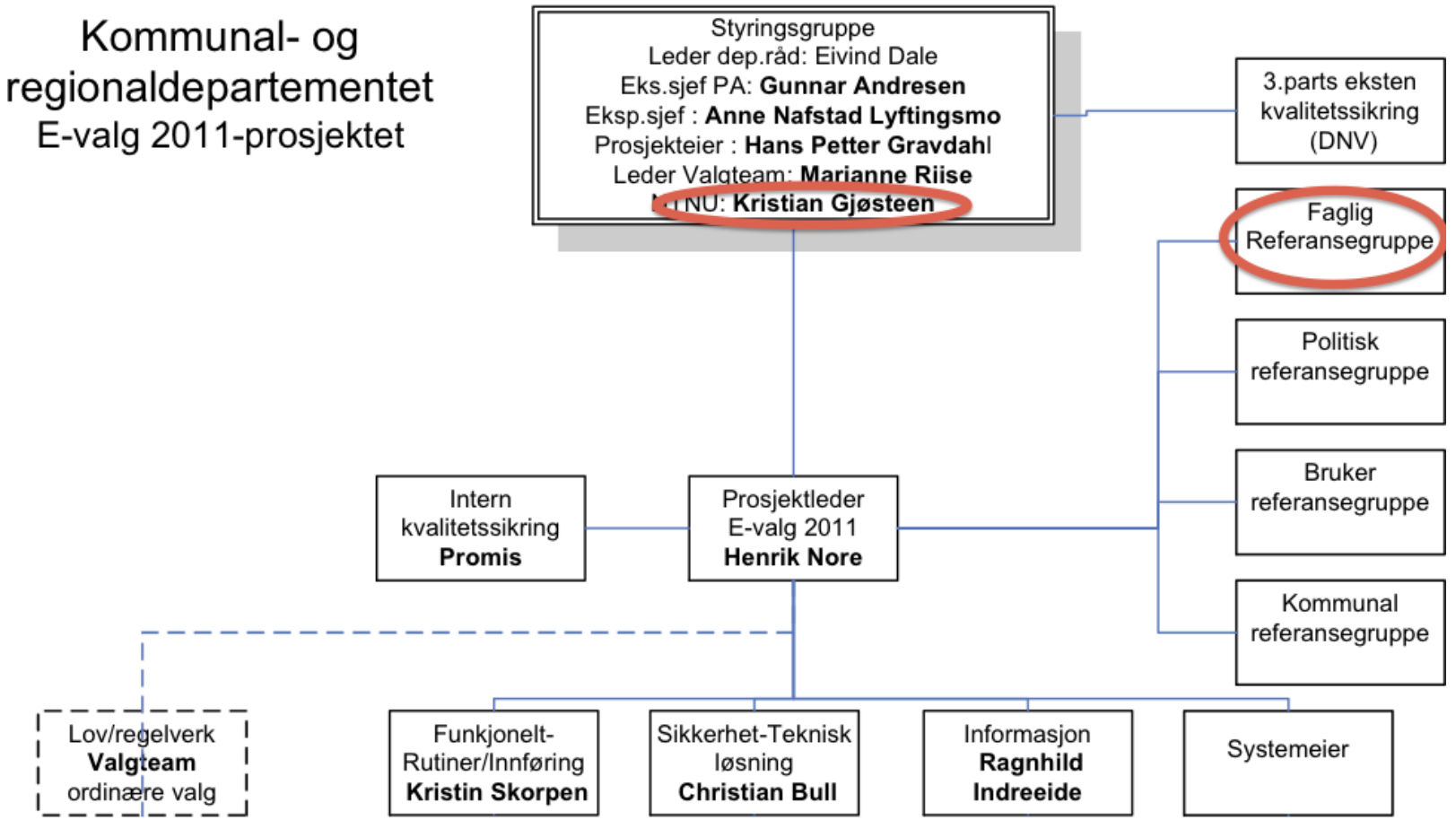
Note II

The project is also about election administration and counting of paper votes.

The E-valg 2011 Project



The E-valg 2011 Project



Assumption: Trust

We trust the government not to conspire against us.

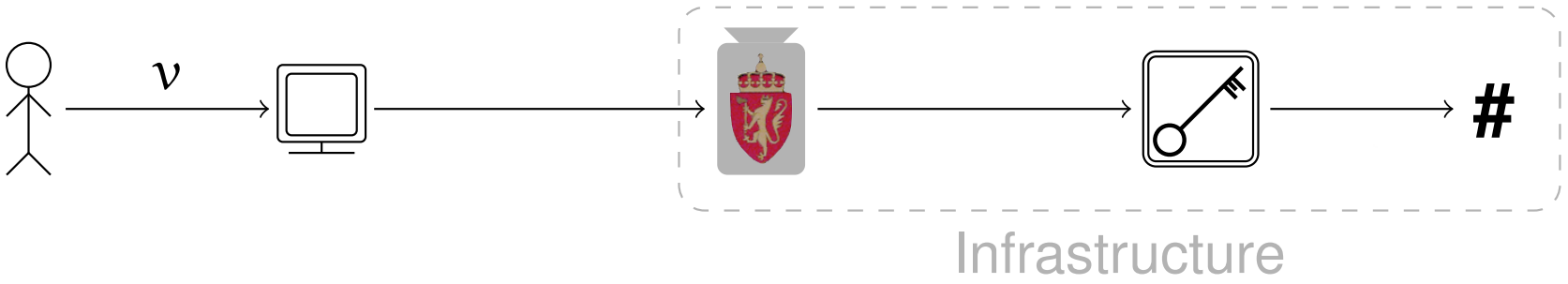
Assumption: Trust

We trust the government not to conspire against us.

But

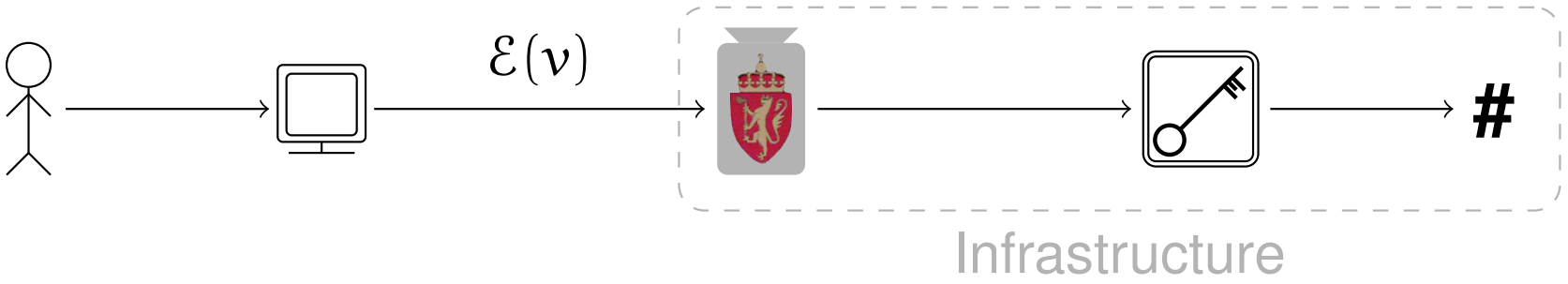
- We do not trust the individuals in government.
- We do not trust to government to do things right.

Internet Elections: Basic Idea



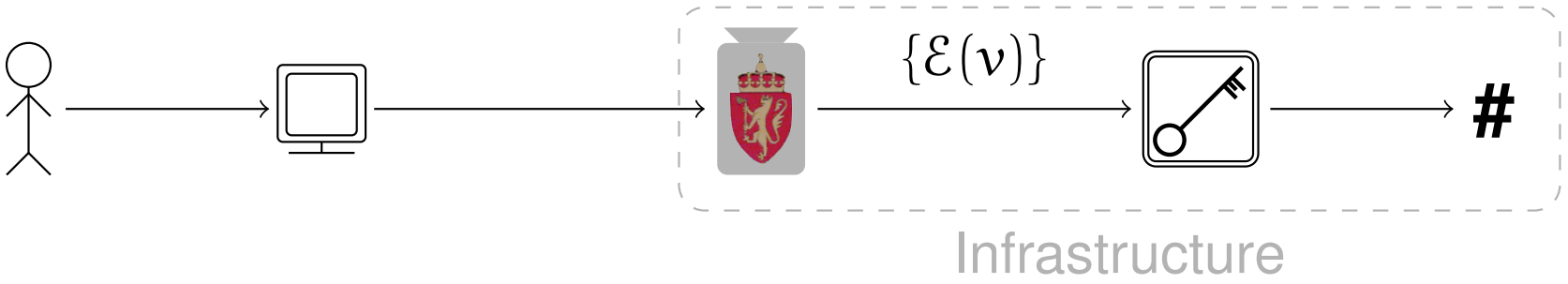
— The voter tells his pc what to vote.

Internet Elections: Basic Idea



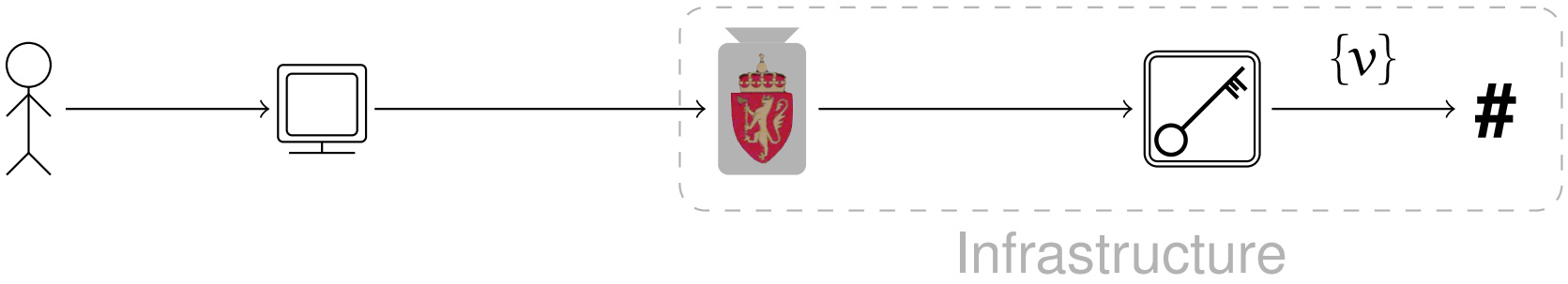
— The pc encrypts the vote and submits it to a ballot box.

Internet Elections: Basic Idea



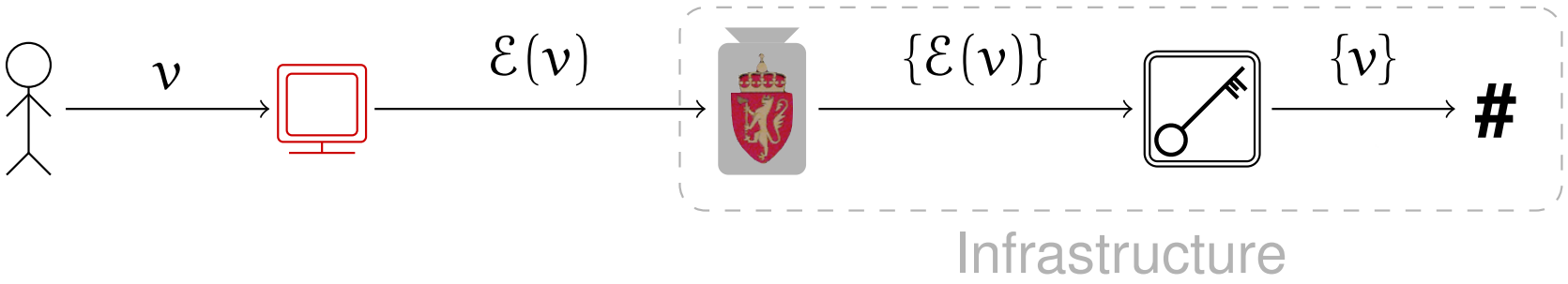
— After voting is done, the ballot box shuffles ciphertexts and submits them for decryption.

Internet Elections: Basic Idea



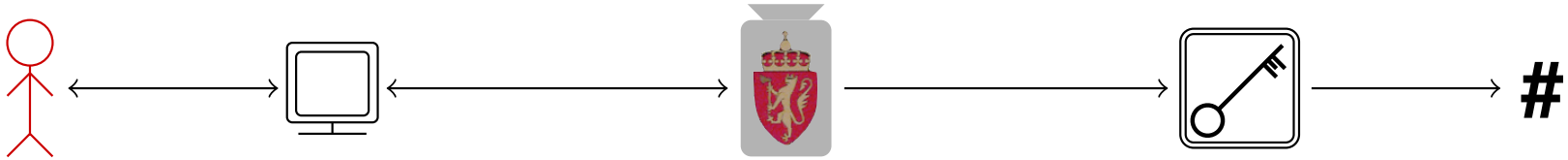
— A decryption service decrypts the ciphertext, shuffles the results and submits the decrypted ballots for counting.

Internet Elections: Basic Idea



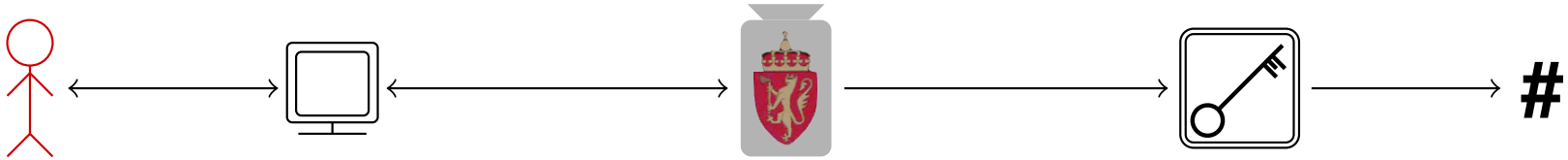
— The Estonian system is very similar to this sketch.

Functional Requirement: Usability



Our voter finds himself participating in a cryptographic protocol.

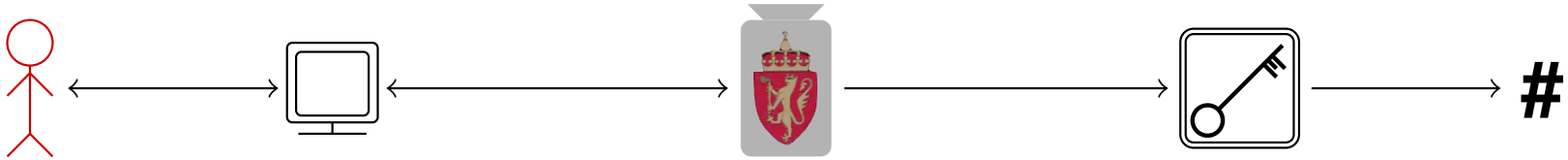
Functional Requirement: Usability



Our voter finds himself participating in a cryptographic protocol.

— Voters won't even do lightweight crypto.

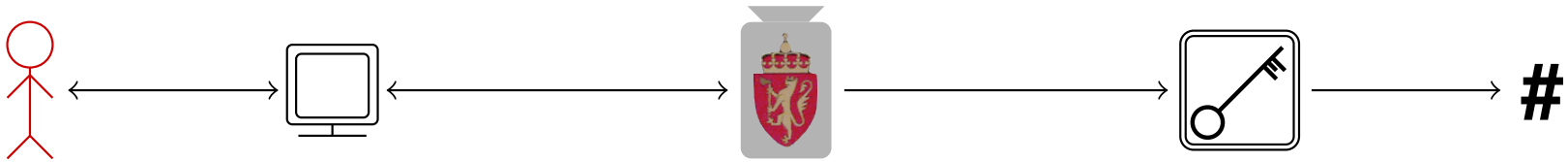
Functional Requirement: Usability



Our voter finds himself participating in a cryptographic protocol.

- Voters won't even do lightweight crypto.
- Voters are subject to «random faults» and possibly also «fault attacks» (e.g. social engineering).

Functional Requirement: Usability



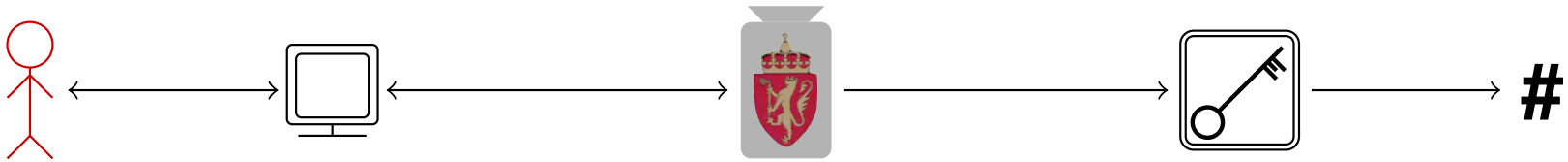
Our voter finds himself participating in a cryptographic protocol.

- Voters won't even do lightweight crypto.
- Voters are subject to «random faults» and possibly also «fault attacks» (e.g. social engineering).

Observation

Our cryptographic protocol has one player that is unreliable and severly constrained.

Functional Requirement: Usability



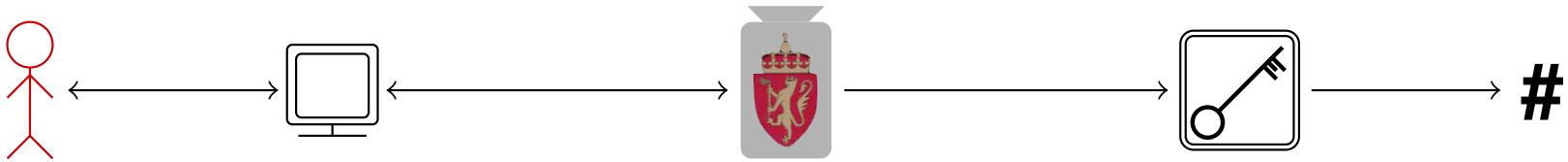
Our voter finds himself participating in a cryptographic protocol.

- Voters won't even do lightweight crypto.
- Voters are subject to «random faults» and possibly also «fault attacks» (e.g. social engineering).

Observation

Training is not an option.

Functional Requirement: Usability



Our voter finds himself participating in a cryptographic protocol.

- Voters won't even do lightweight crypto.
- Voters are subject to «random faults» and possibly also «fault attacks» (e.g. social engineering).

It must be easy for almost all voters to use the protocol correctly.

Security Requirements

Privacy A voter's ballot should remain secret (or as secret as possible).

Integrity The integrity of the entire ballot should be ensured from submission to counting (most voters should be able to verify integrity).

Buying Votes A vote buyer/stealer should have no method to ensure that votes stay bought/stolen

Security Requirements

Privacy A voter's ballot should remain secret (or as secret as possible).

Integrity The integrity of the entire ballot should be ensured from submission to counting (most voters should be able to verify integrity).

Buying Votes A vote buyer/stealer should have no method to ensure that votes stay bought/stolen

Note

The entire ballot matters.

Security Requirements

Privacy A voter's ballot should remain secret (or as secret as possible).

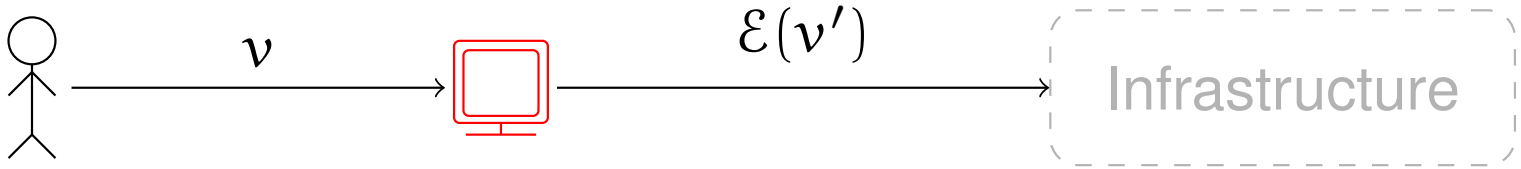
Integrity The integrity of the entire ballot should be ensured from submission to counting (most voters should be able to verify integrity).

Buying Votes A vote buyer/stealer should have no method to ensure that votes stay bought/stolen

Note

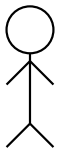
Universal verifiability is hard!

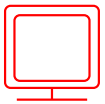
Problem: Compromised Computers



— In Estonia, a compromised computer means no privacy or integrity. The computer may also vote undetectably on your behalf.

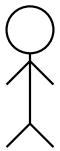
Problem: Compromised Computers

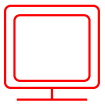




— Compromised computers were considered by Chaum's *SureVote* (2000) and UK's CESG's *e-Voting Security Study* (2002).

Problem: Compromised Computers

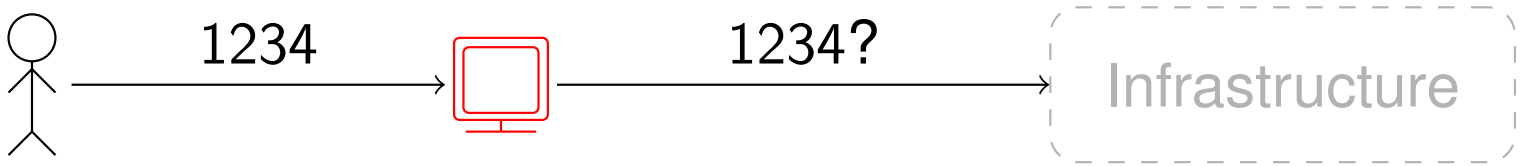




— Every personal voting card has a unique table:

Candidate	Vote Code	Receipt Code
A	1234	5678
B	4321	0987

Problem: Compromised Computers

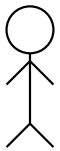


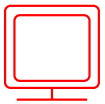
— Every personal voting card has a unique table:

Candidate	Vote Code	Receipt Code
A	1234	5678
B	4321	0987

— To vote for Candidate A, the voter gives the vote code 1234 to the computer, which passes it on to the infrastructure.

Problem: Compromised Computers



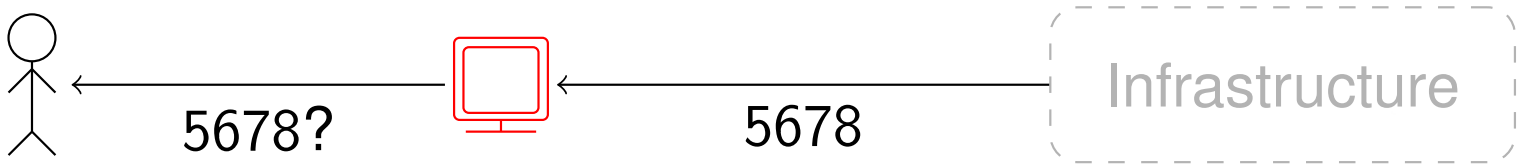


— Every personal voting card has a unique table:

Candidate	Vote Code	Receipt Code
A	1234	5678
B	4321	0987

— The infrastructure essentially (details vary) has a copy of the voter's table and records a vote for Candidate A.

Problem: Compromised Computers

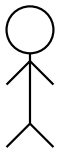


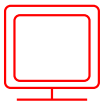
— Every personal voting card has a unique table:

Candidate	Vote Code	Receipt Code
A	1234	5678
B	4321	0987

— The receipt code is sent to the computer, which sends it to the voter, who verifies it.

Problem: Compromised Computers

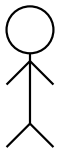


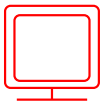


— Compromised computers were considered by Chaum's *SureVote* (2000) and UK's CESG's *e-Voting Security Study* (2002).

Question: Can these systems be adapted for Norway?

Problem: Compromised Computers



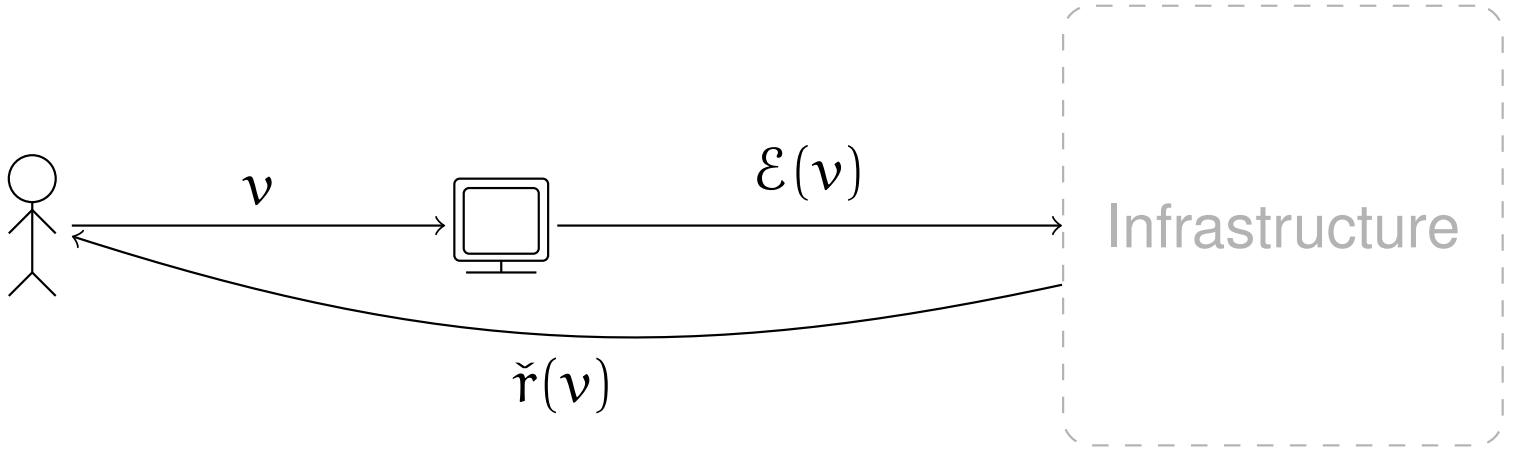


— Compromised computers were considered by Chaum's *SureVote* (2000) and UK's CESG's *e-Voting Security Study* (2002).

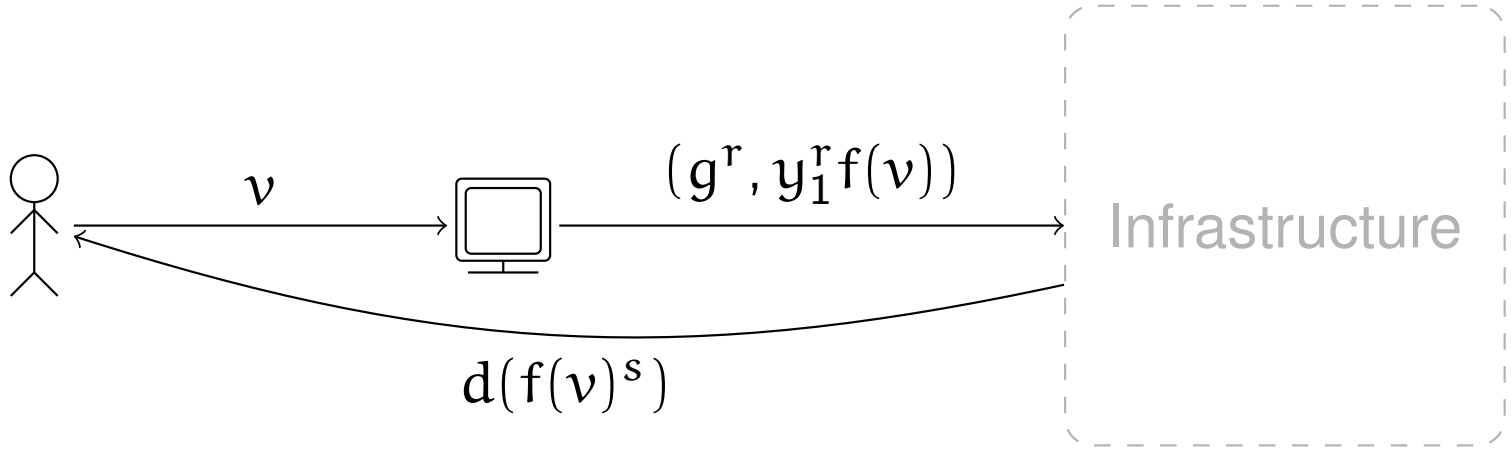
Question: Can these systems be adapted for Norway?

Electronic voting should not change voting patterns.

Computing the Receipt Code

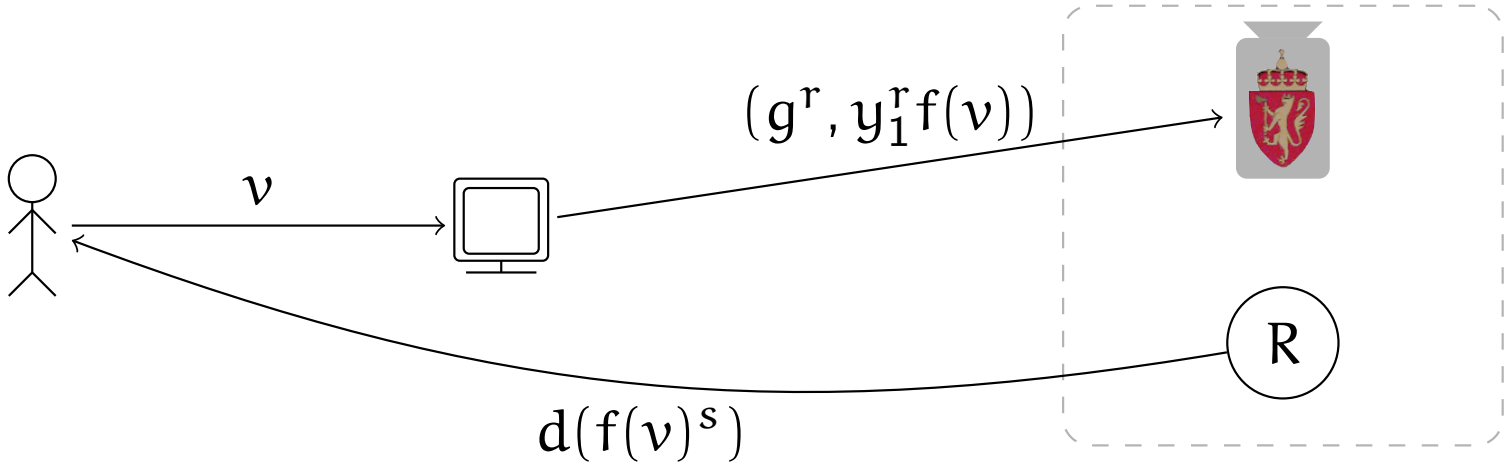


Computing the Receipt Code



- f encoding function
- s per-voter exponent, random
- d $G \rightarrow$ {receipt code set}, per-voter, pseudo-random
- y_1 election encryption key

Computing the Receipt Code



- f encoding function
- s per-voter exponent, random
- d $G \rightarrow$ {receipt code set}, per-voter, pseudo-random
- y_1 election encryption key

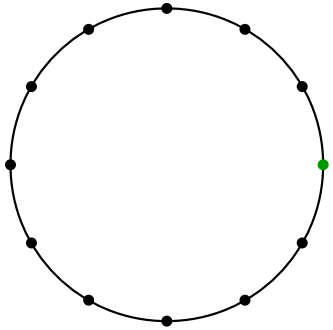
ElGamal, Geometry and Donuts

Here's a different perspective on ElGamal encryption.

Note

These donuts aren't related to elliptic curve donuts.

ElGamal, Geometry and Donuts

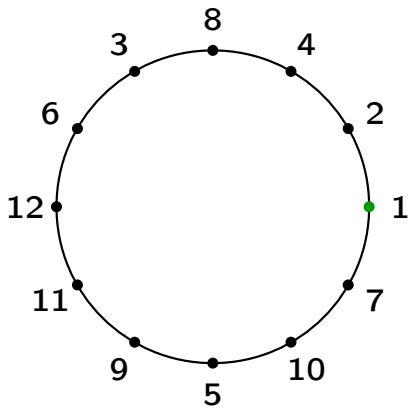


We think of a finite cyclic group $\mathbb{G}$ as a bunch of elements arranged on a circle such that the group operation corresponds to angle addition.

Note

Angle multiples corresponds to exponentiations in the group.

ElGamal, Geometry and Donuts

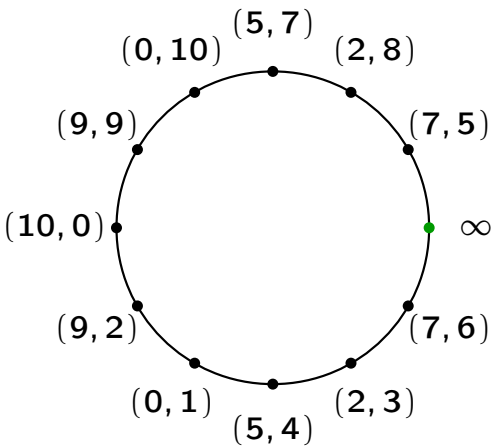


We think of a finite cyclic group G as a bunch of elements arranged on a circle such that the group operation corresponds to angle addition.

Note

Angle multiples corresponds to exponentiations in the group.

ElGamal, Geometry and Donuts

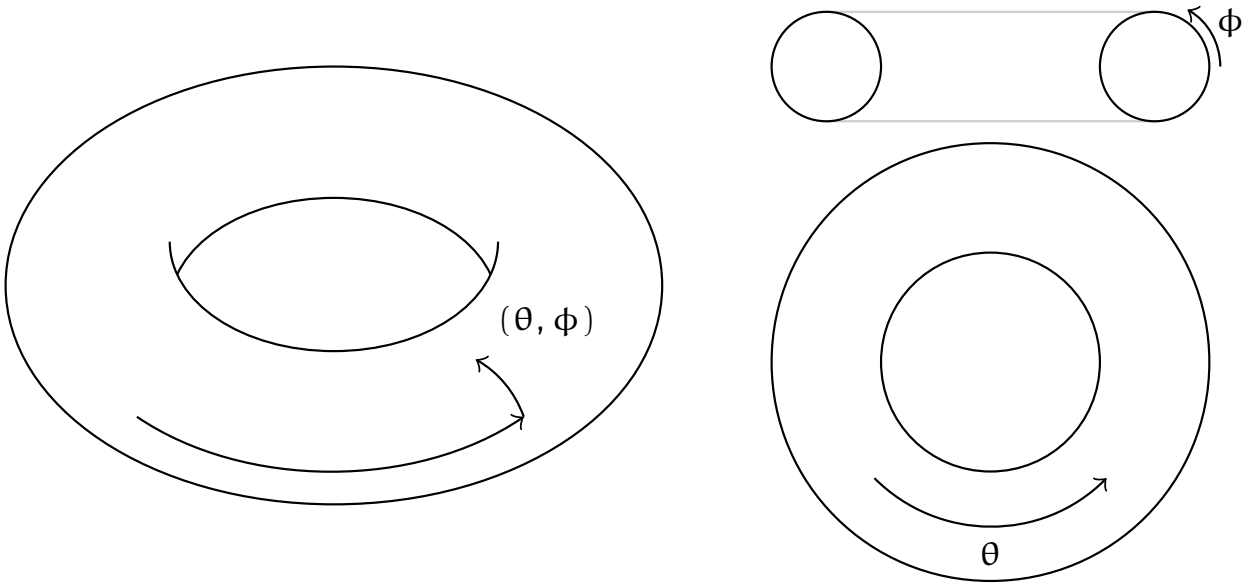


We think of a finite cyclic group G as a bunch of elements arranged on a circle such that the group operation corresponds to angle addition.

Note

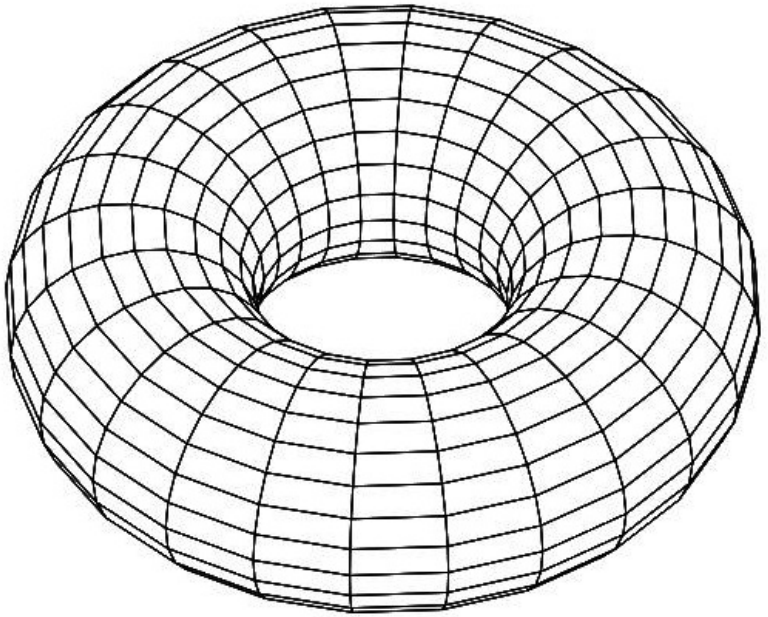
Angle multiples corresponds to exponentiations in the group.

ElGamal, Geometry and Donuts



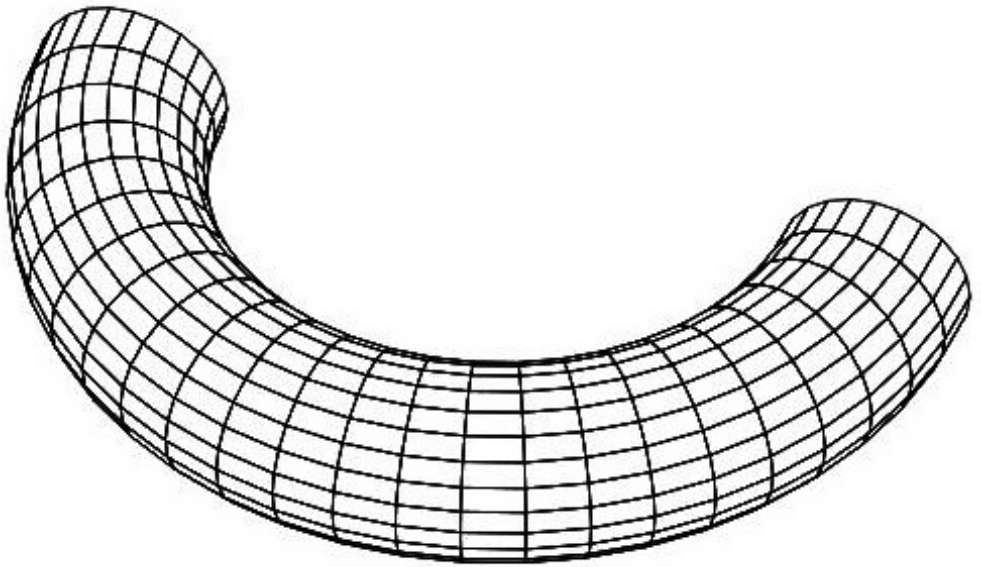
If a group element corresponds to an angle, a pair of group elements corresponds to a point of the surface of a torus.

ElGamal, Geometry and Donuts



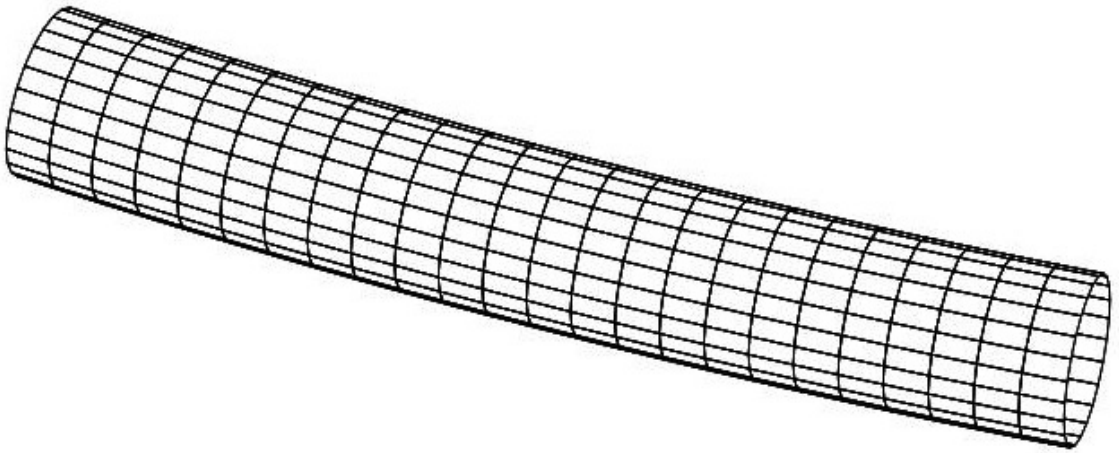
The surface of a torus is essentially a plane, just like the ordinary plane.

ElGamal, Geometry and Donuts



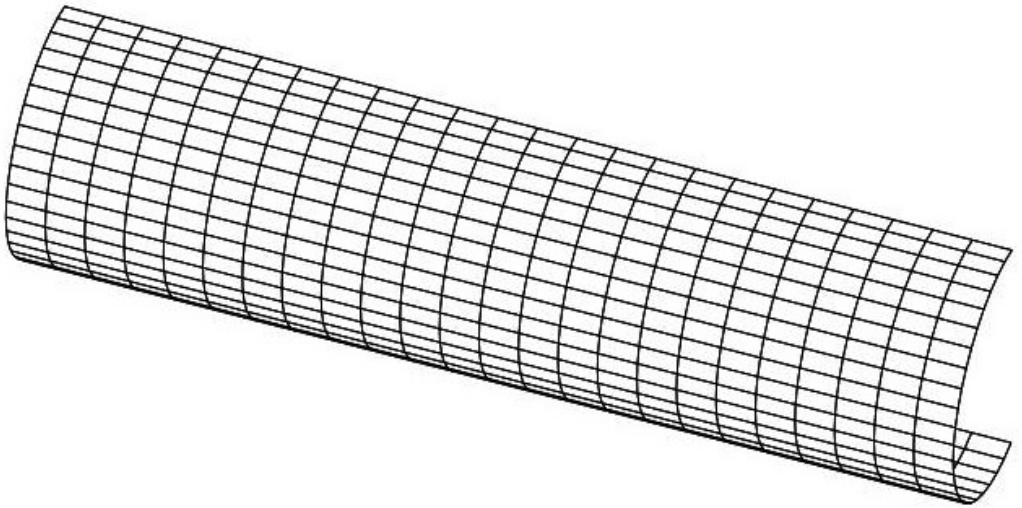
The surface of a torus is essentially a plane, just like the ordinary plane.

ElGamal, Geometry and Donuts



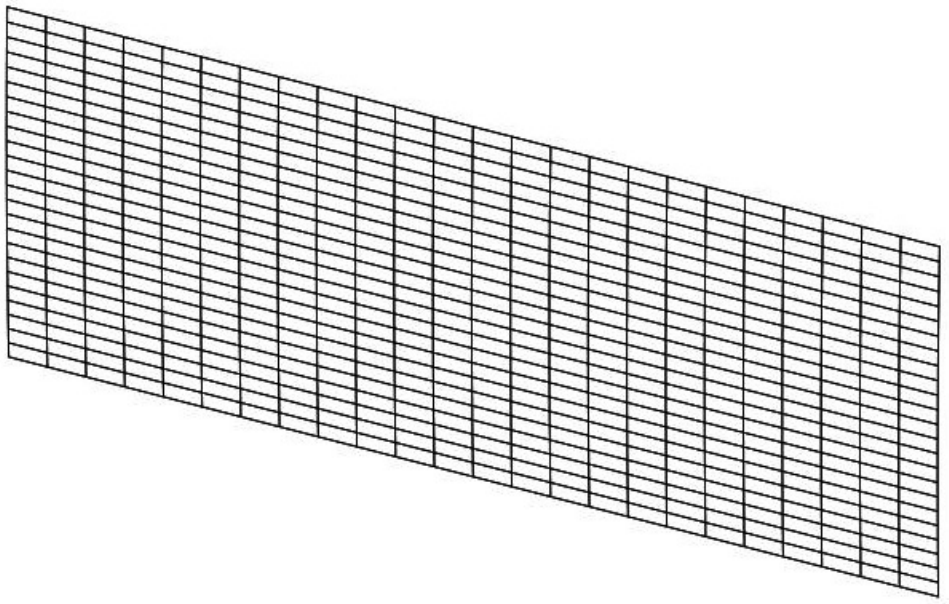
The surface of a torus is essentially a plane, just like the ordinary plane.

ElGamal, Geometry and Donuts



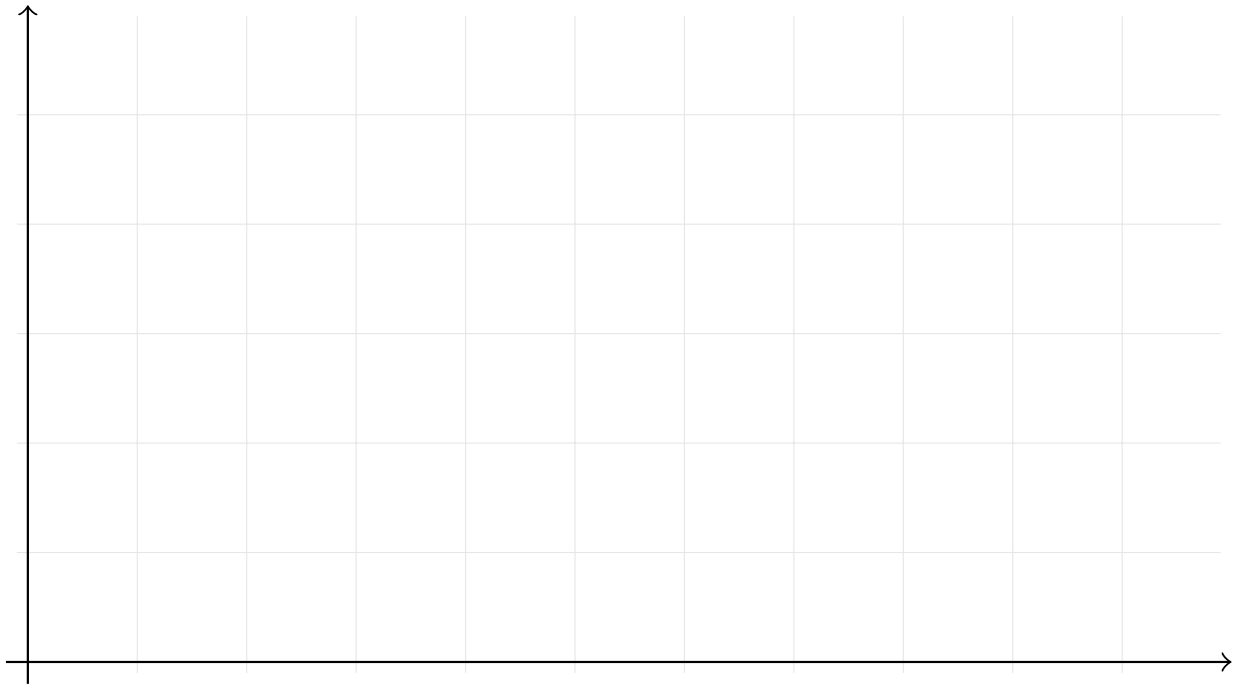
The surface of a torus is essentially a plane, just like the ordinary plane.

ElGamal, Geometry and Donuts



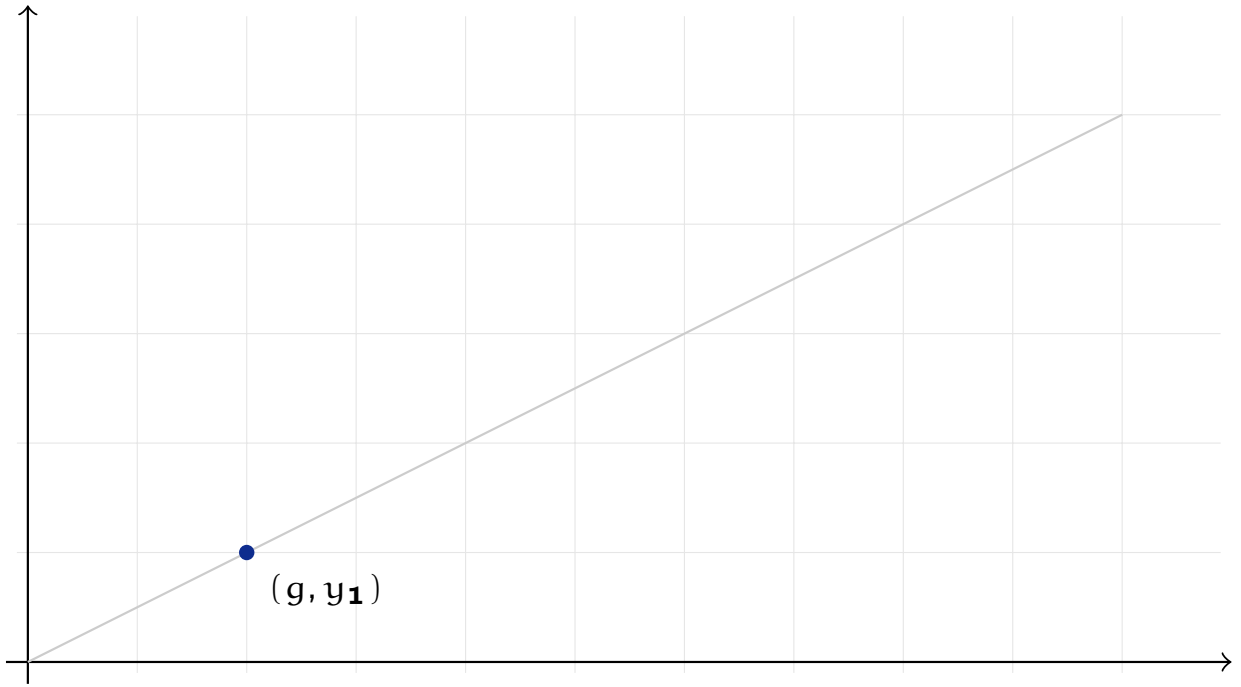
The surface of a torus is essentially a plane, just like the ordinary plane.

ElGamal, Geometry and Donuts



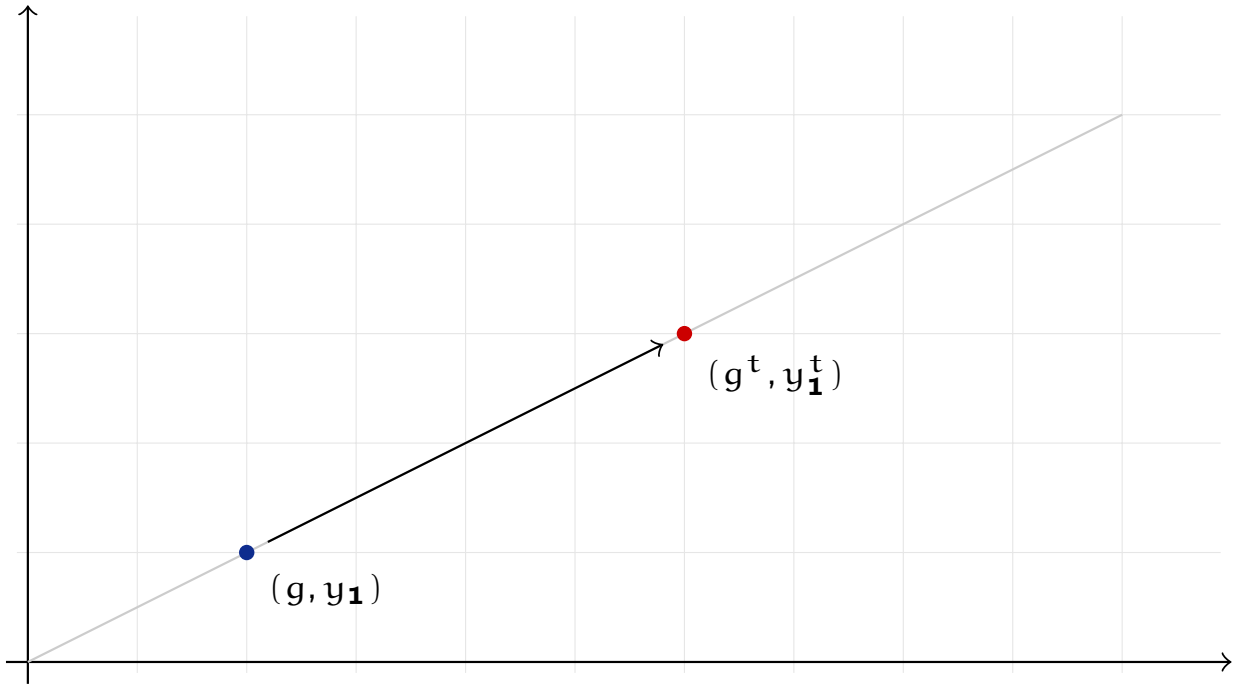
In other words: We think of pairs of group elements as points in an ordinary plane.

ElGamal, Geometry and Donuts



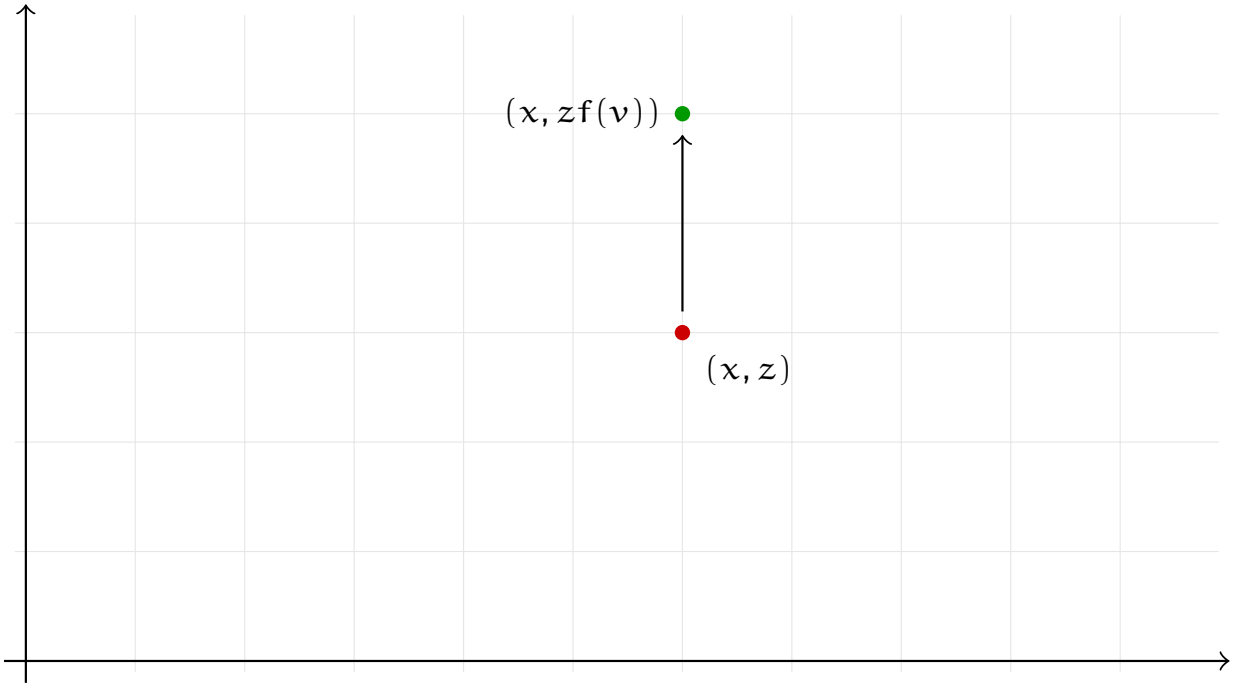
The generator g and the encryption key y_1 describe a point and a line through the origin.

ElGamal, Geometry and Donuts



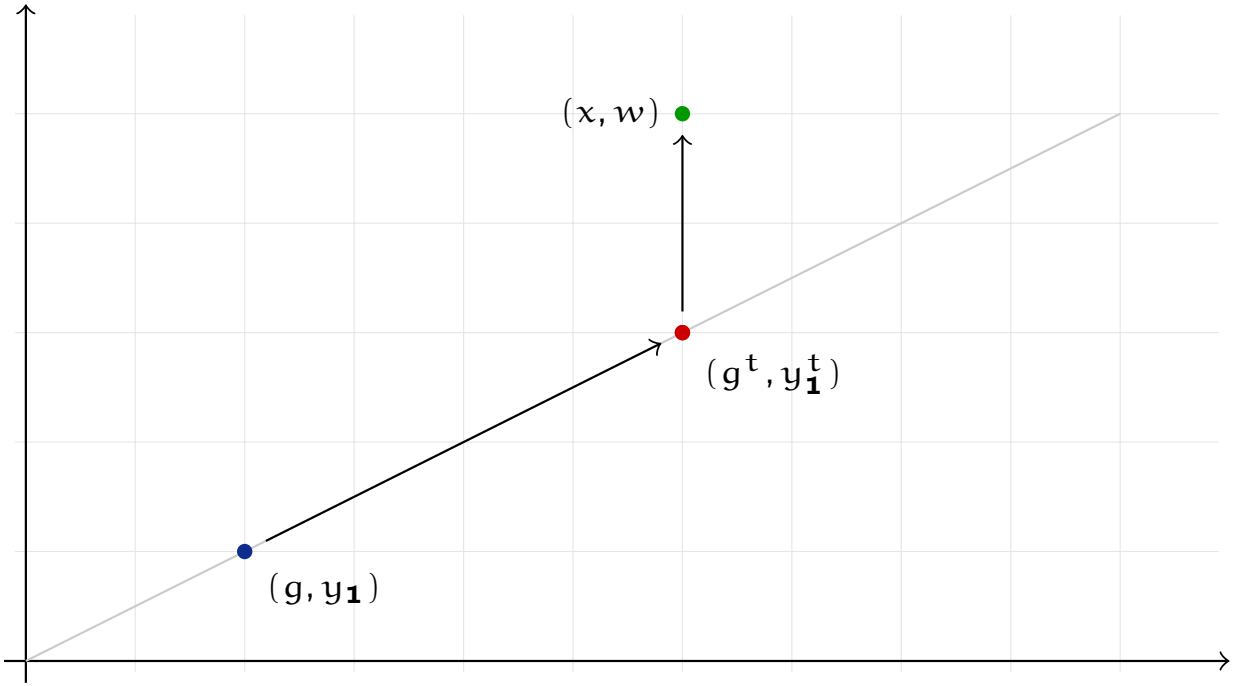
For any integer t, the points (g, y_1) and (g^t, y_1^t) lie on the same line through the origin.

ElGamal, Geometry and Donuts



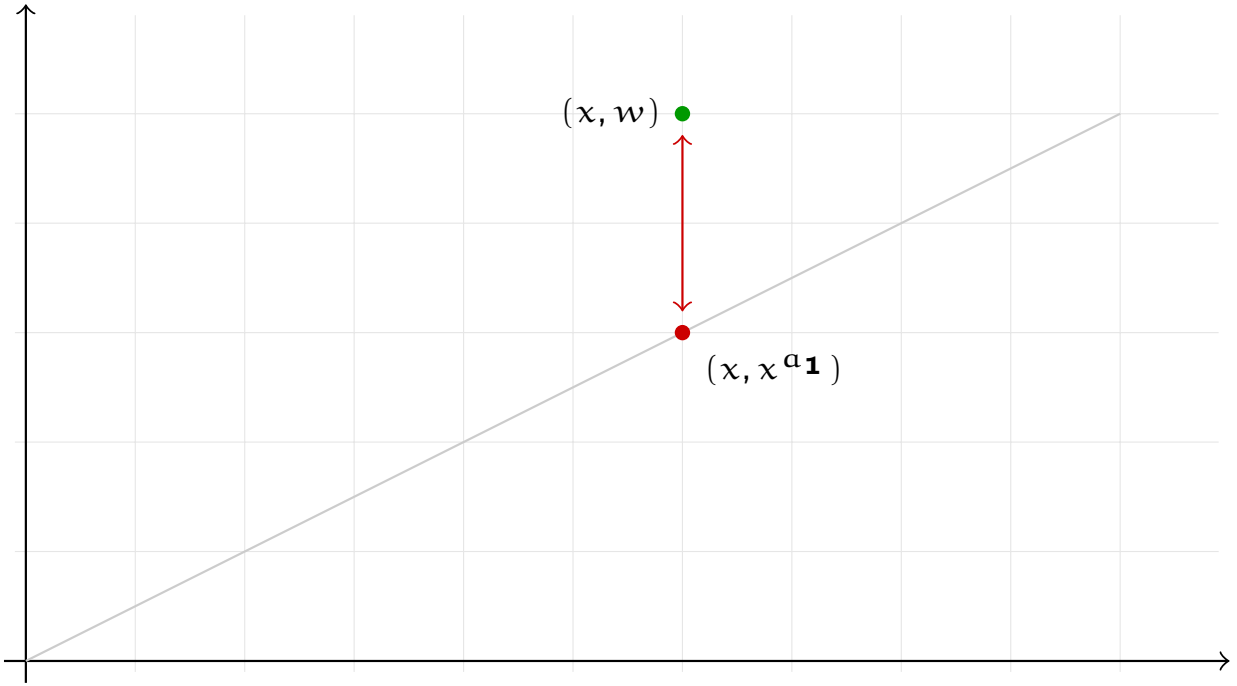
Multiplying $f(v)$ into the second coordinate amounts to a vertical shift.

ElGamal, Geometry and Donuts



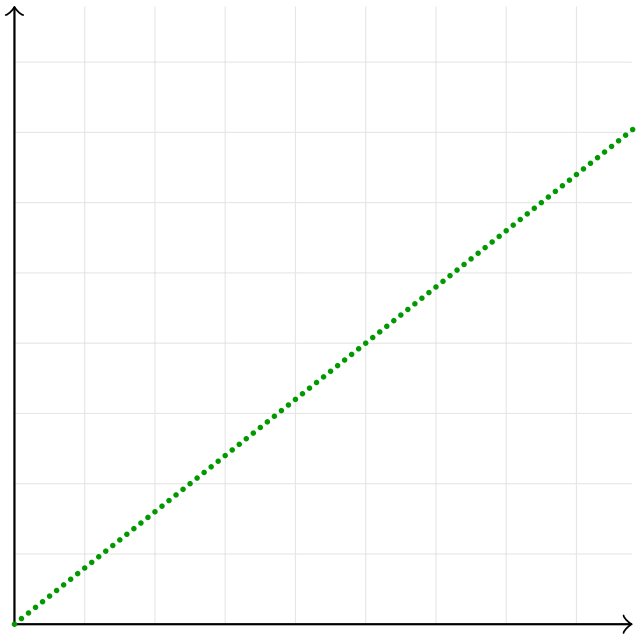
ElGamal encryption: (i) choose a random point on the line defined by the encryption key, and (ii) shift the point vertically by the message.

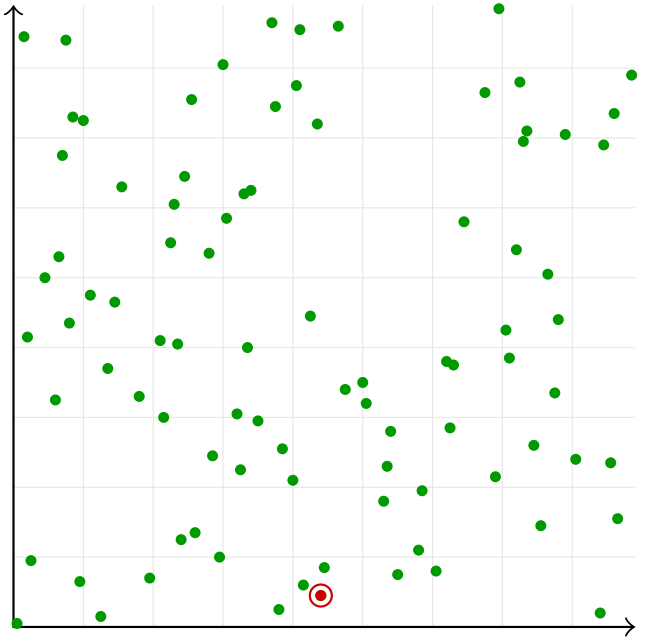
ElGamal, Geometry and Donuts



ElGamal decryption: The decryption key is the slope a_1 of the line. With the slope, we can compute the vertical distance between the ciphertext and the line.

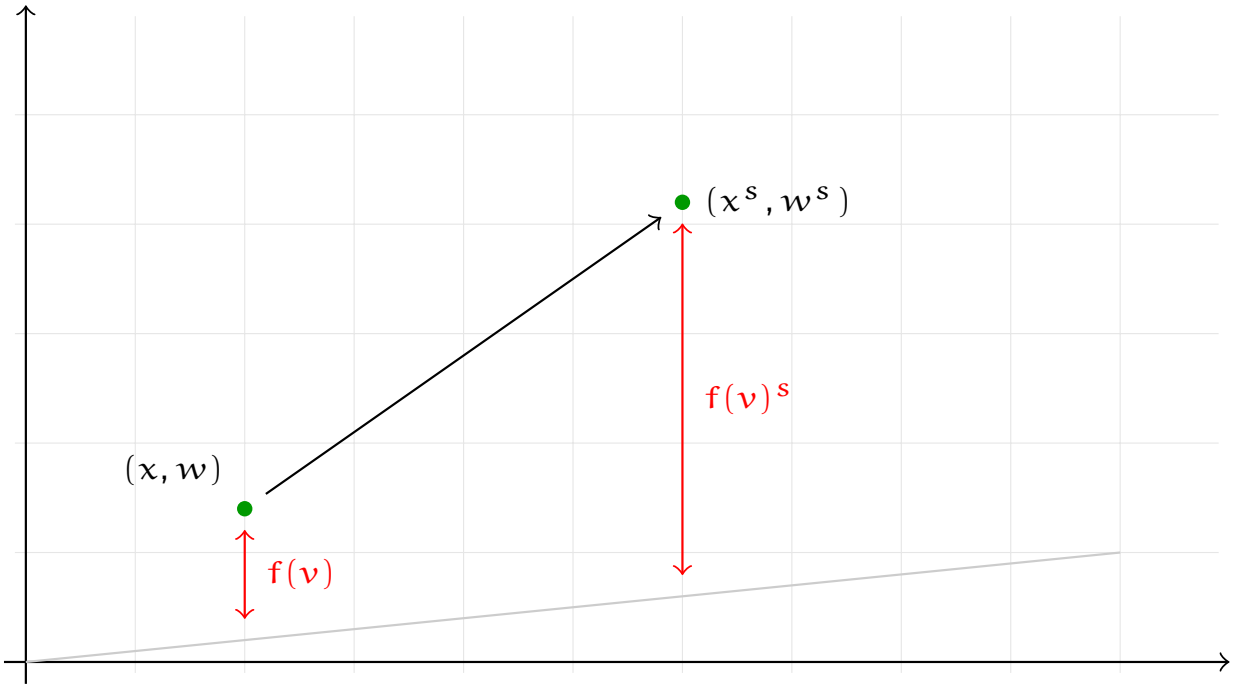
ElGamal, Geometry and Donuts





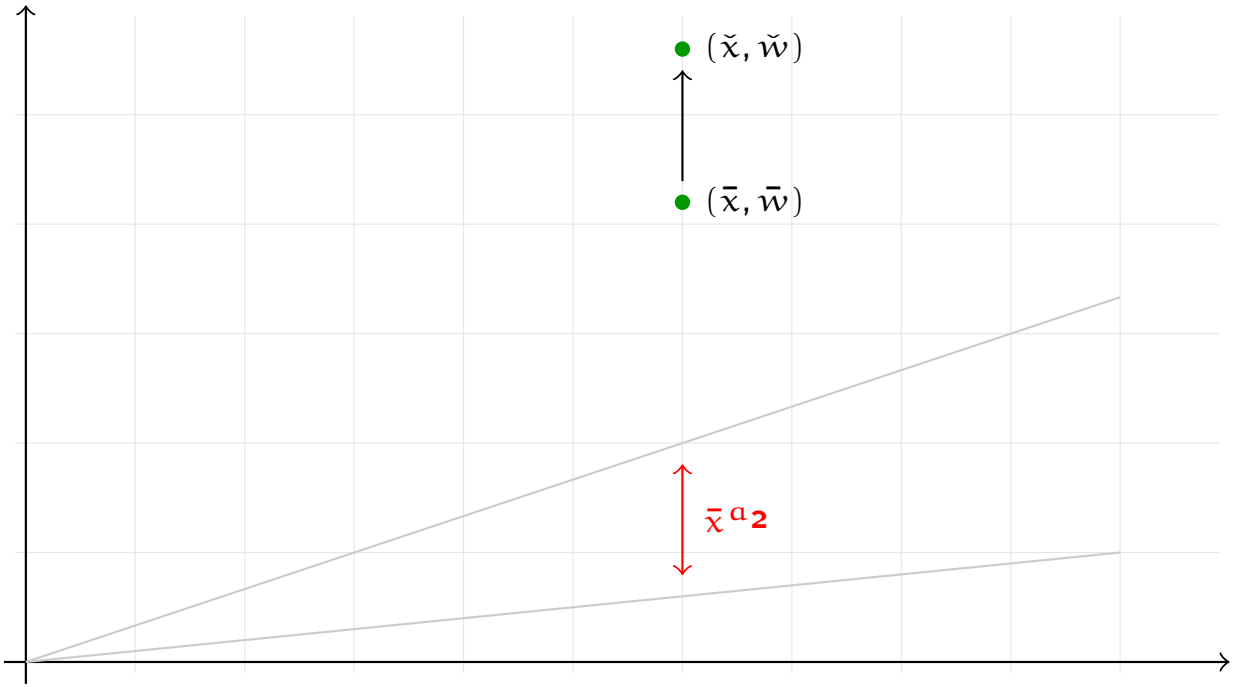
Decision Diffie-Hellman Assumption: Given two points, it is hard to say anything about the vertical distance between the second point and the line defined by the first point, unless you know the slope of the line.

ElGamal, Geometry and Donuts



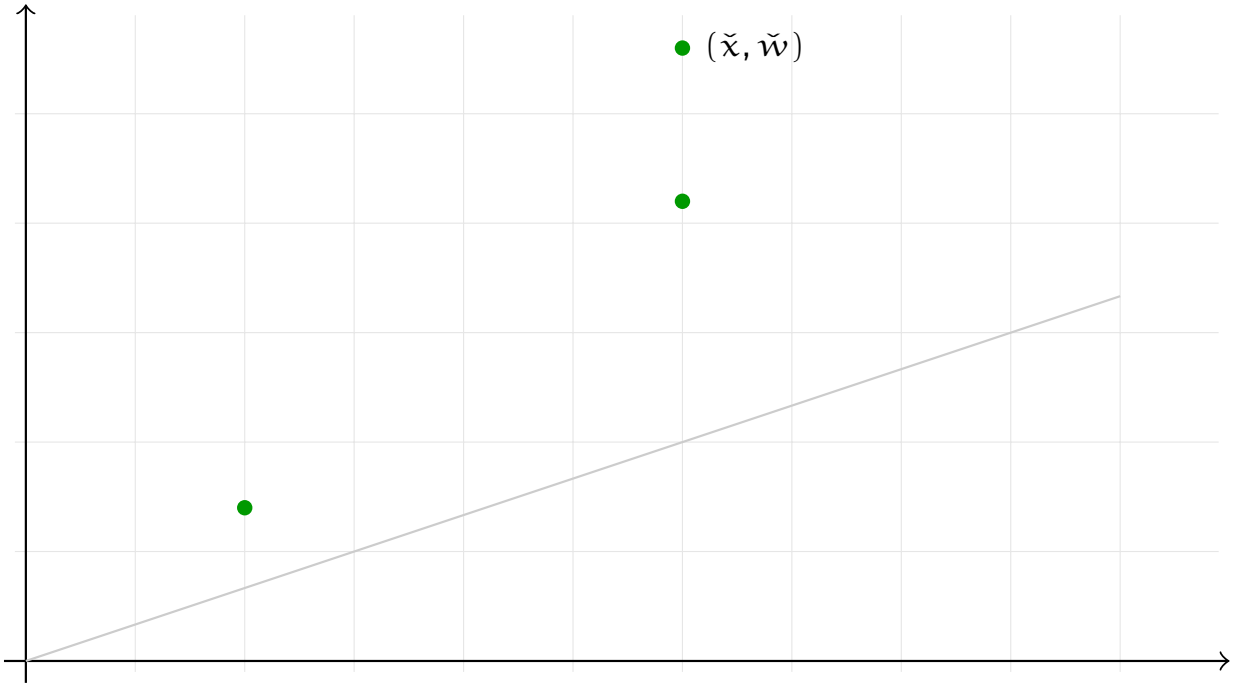
The ballot box knows the per-voter exponent s and computes a new ciphertext $(\bar{x}, \bar{w}) = (x^s, w^s)$.

ElGamal, Geometry and Donuts



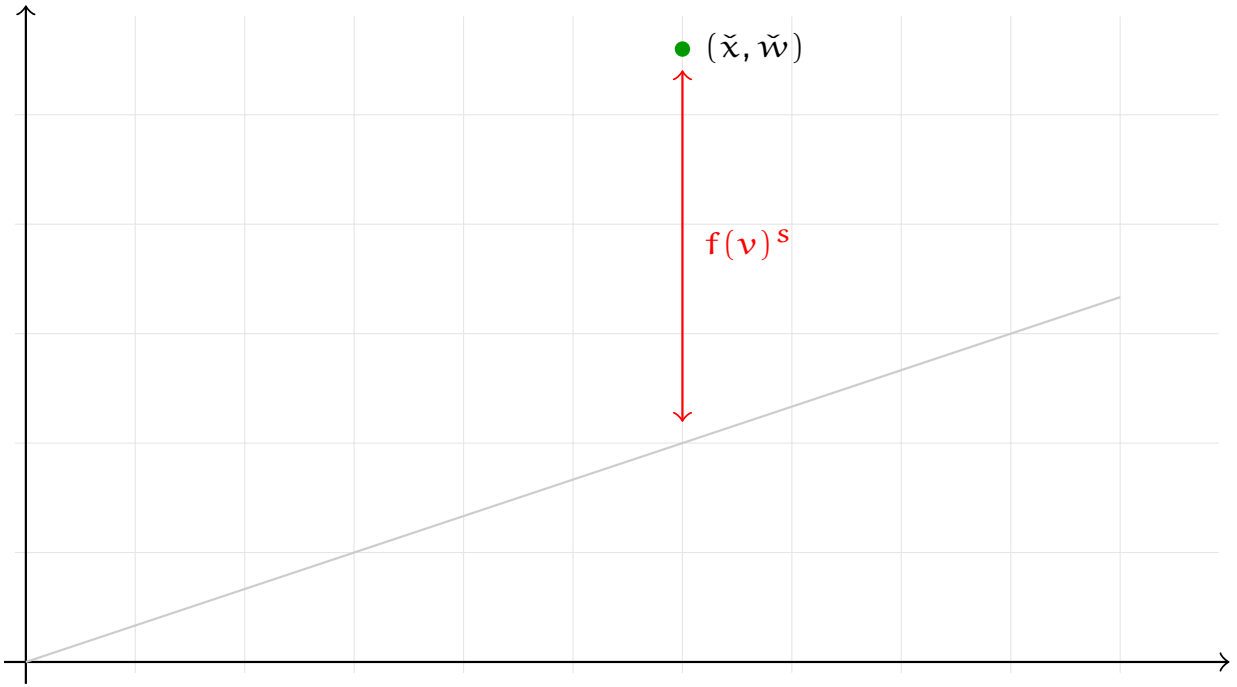
The ballot box knows the *difference* a_2 between two slopes. It uses this to compute a new ciphertext $(\check{x}, \check{w}) = (\bar{x}, \bar{w}\bar{x}^{a_2})$.

ElGamal, Geometry and Donuts



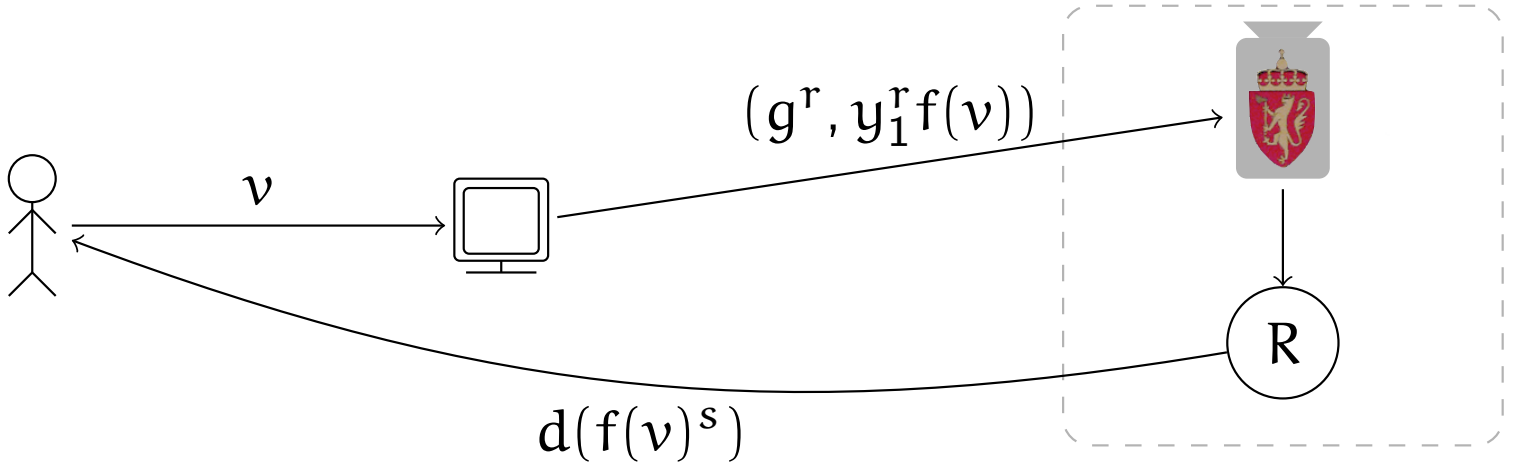
The receipt generator gets all three ciphertexts.

ElGamal, Geometry and Donuts



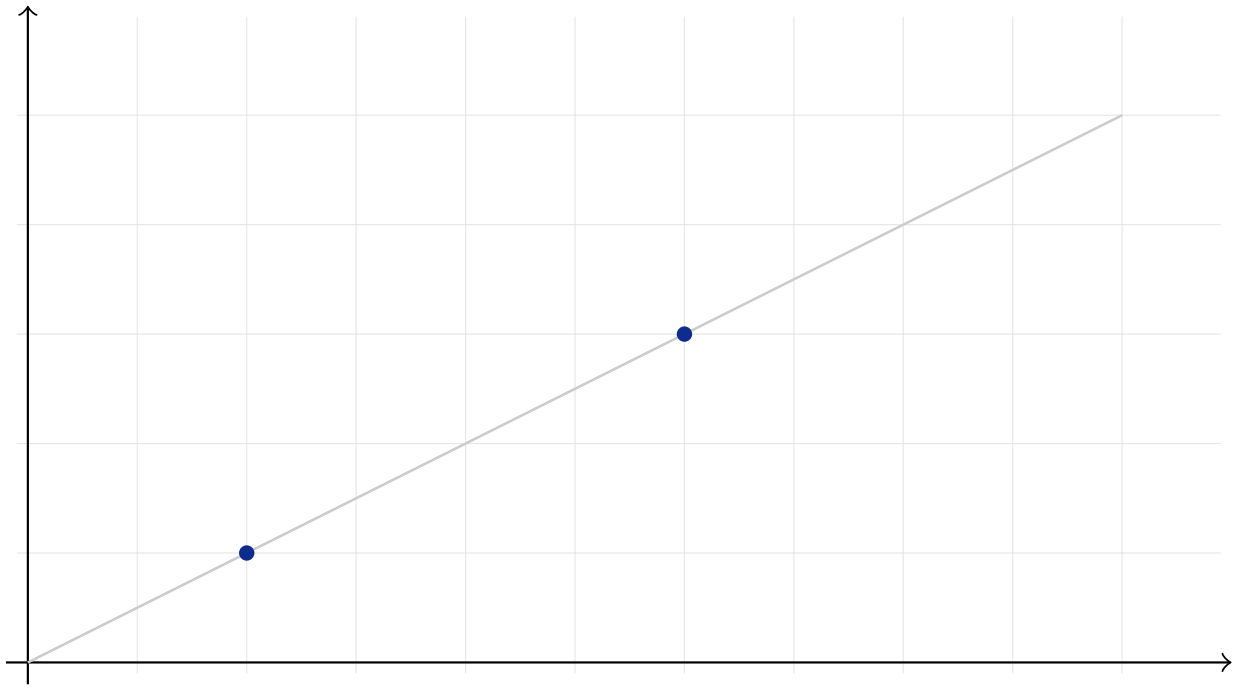
The receipt generator knows the slope of the second line and can decrypt only the final ciphertext.

Computing the Receipt Code



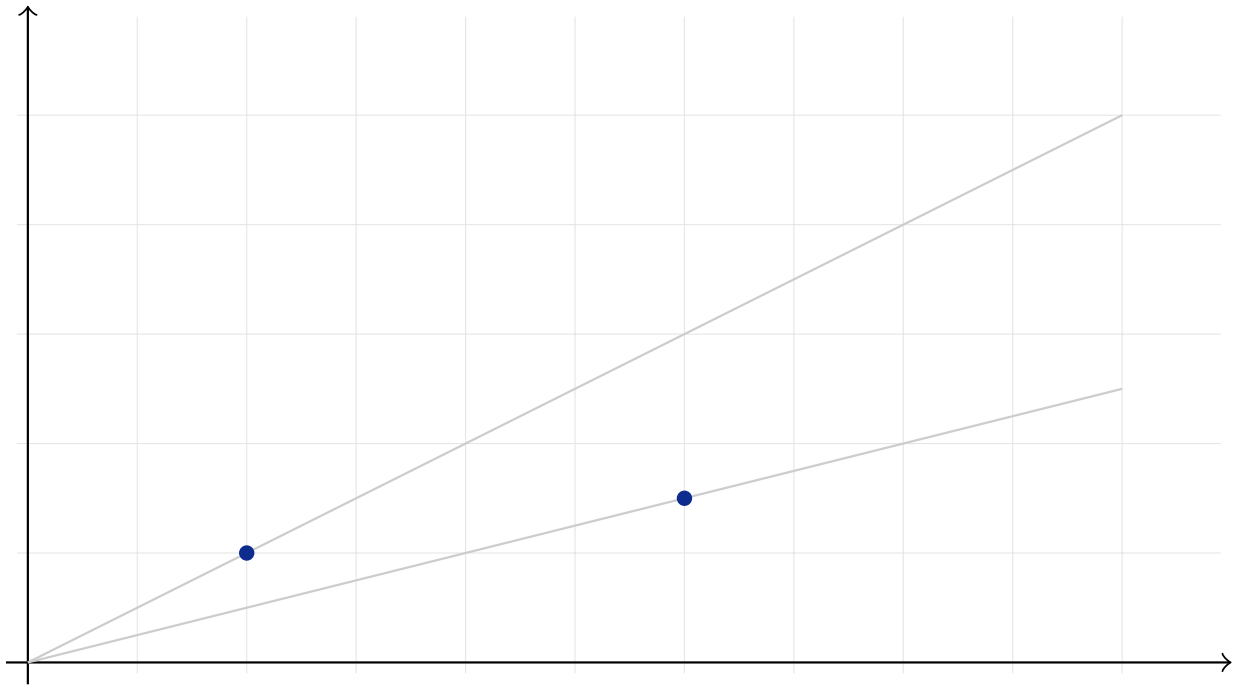
Security: (i) The ballot box sees nothing because of ElGamal, and (ii) the receipt generator sees nothing because $f(v)^s$ looks random. Does it?

Encoding



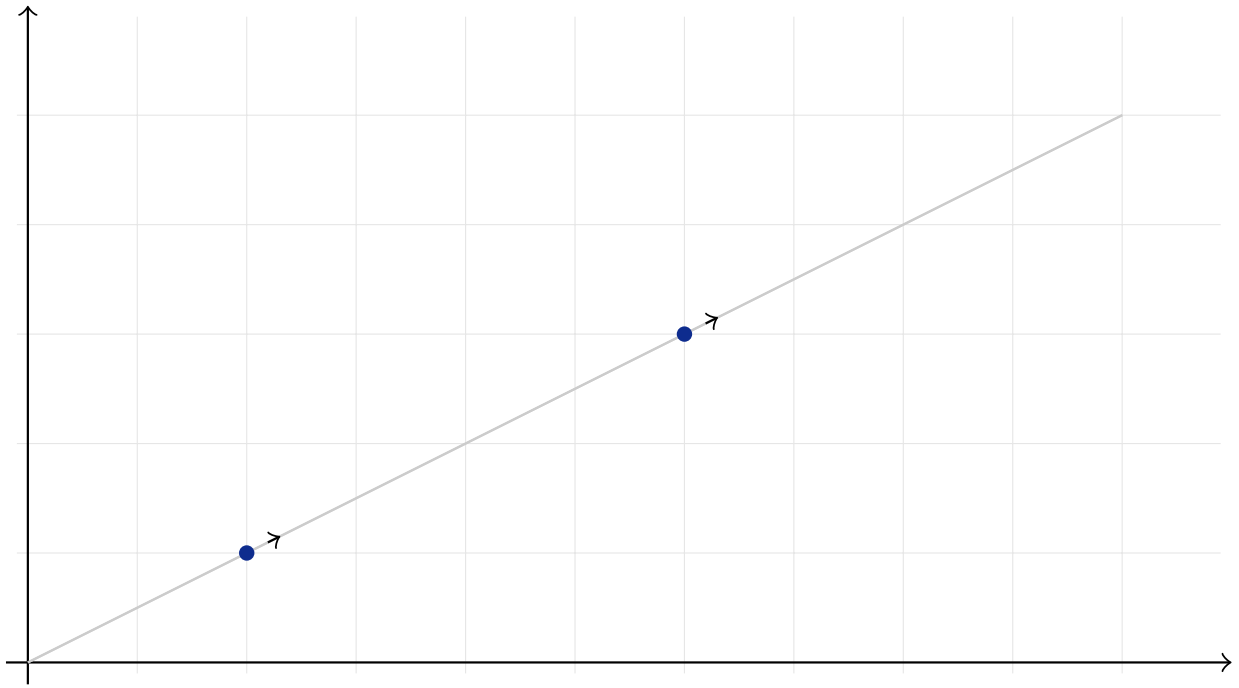
Given two points in the plane, DDH says it is hard to tell if they lie on the same line through the origin.

Encoding



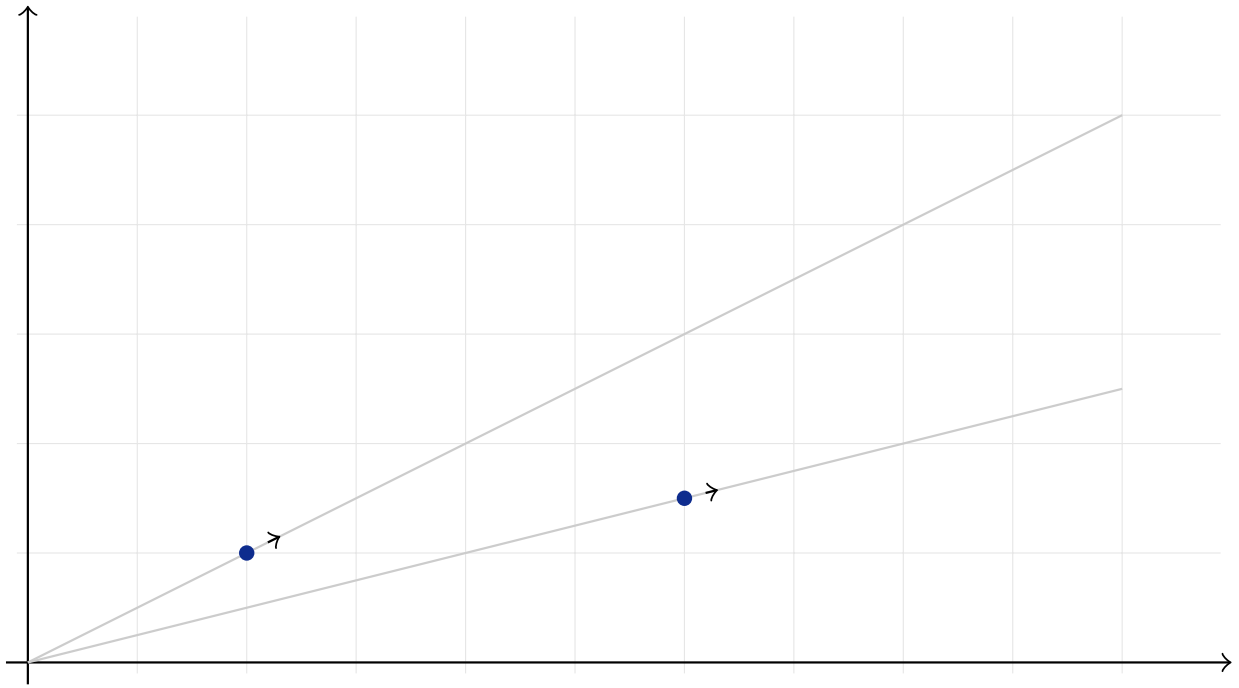
Given two points in the plane, DDH says it is hard to tell if they lie on the same line through the origin, or on different lines.

Encoding



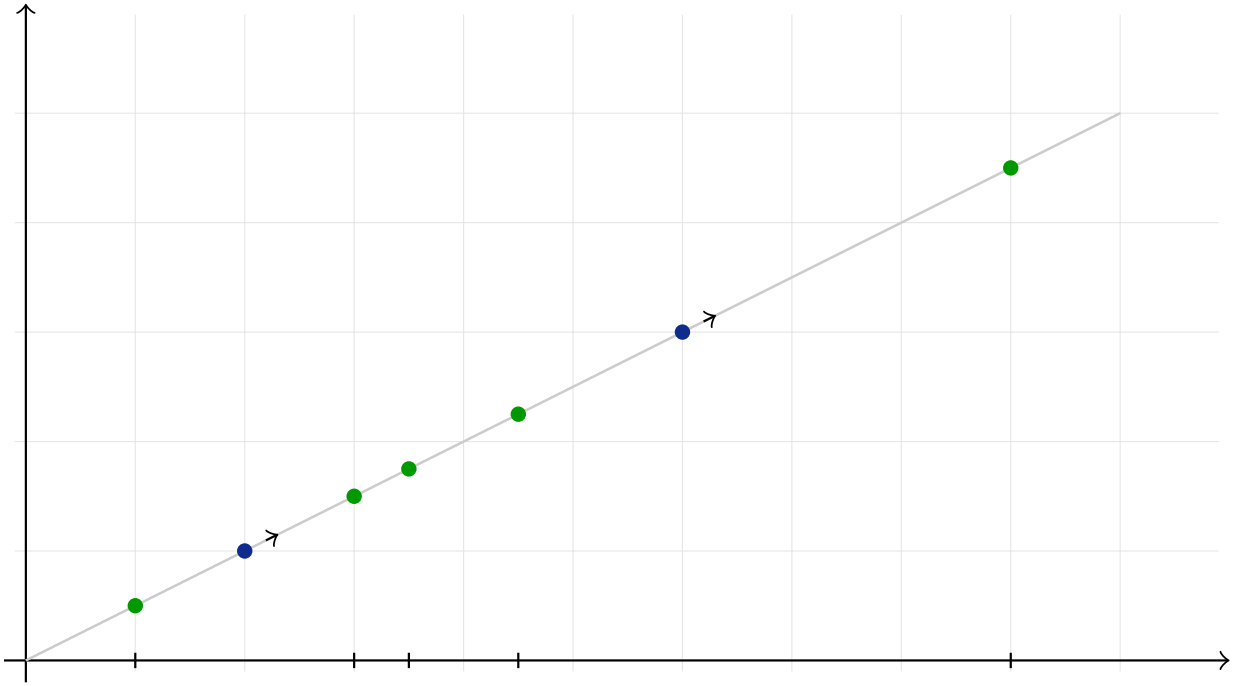
We can make random linear combinations of two points. If the points lie on the same line, the linear combination stays on the line.

Encoding



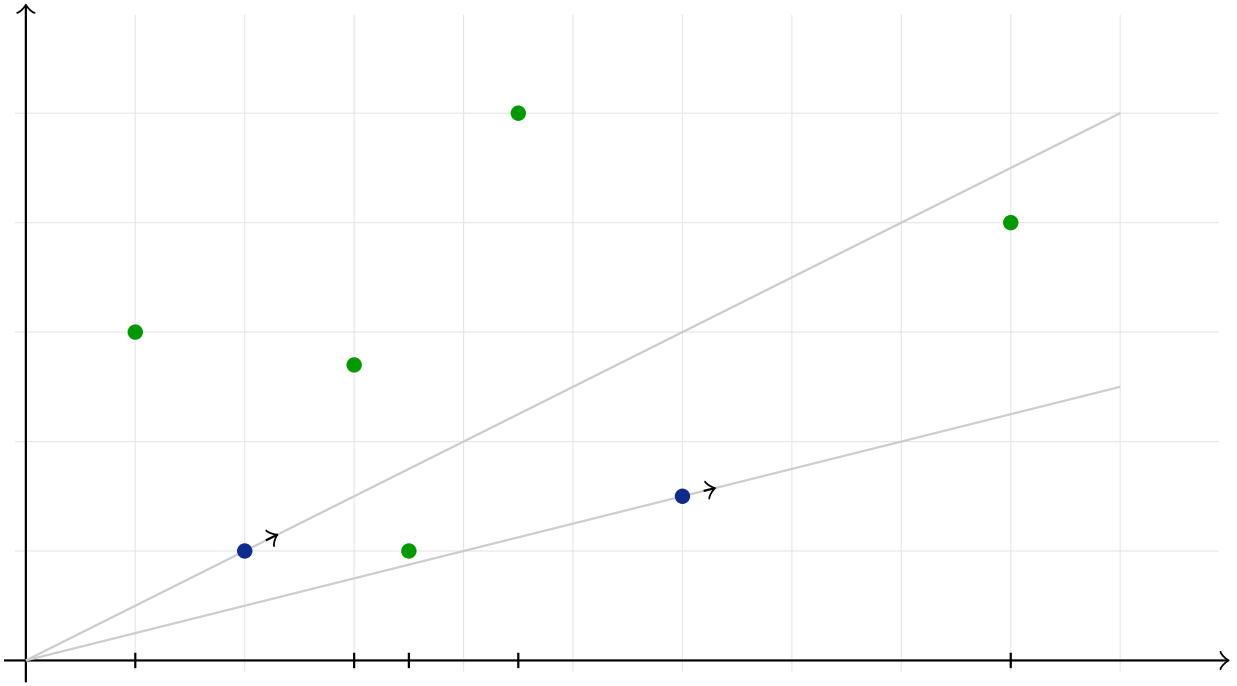
We can make random linear combinations of two points. If the points lie on different lines, the linear combination could end up anywhere.

Encoding



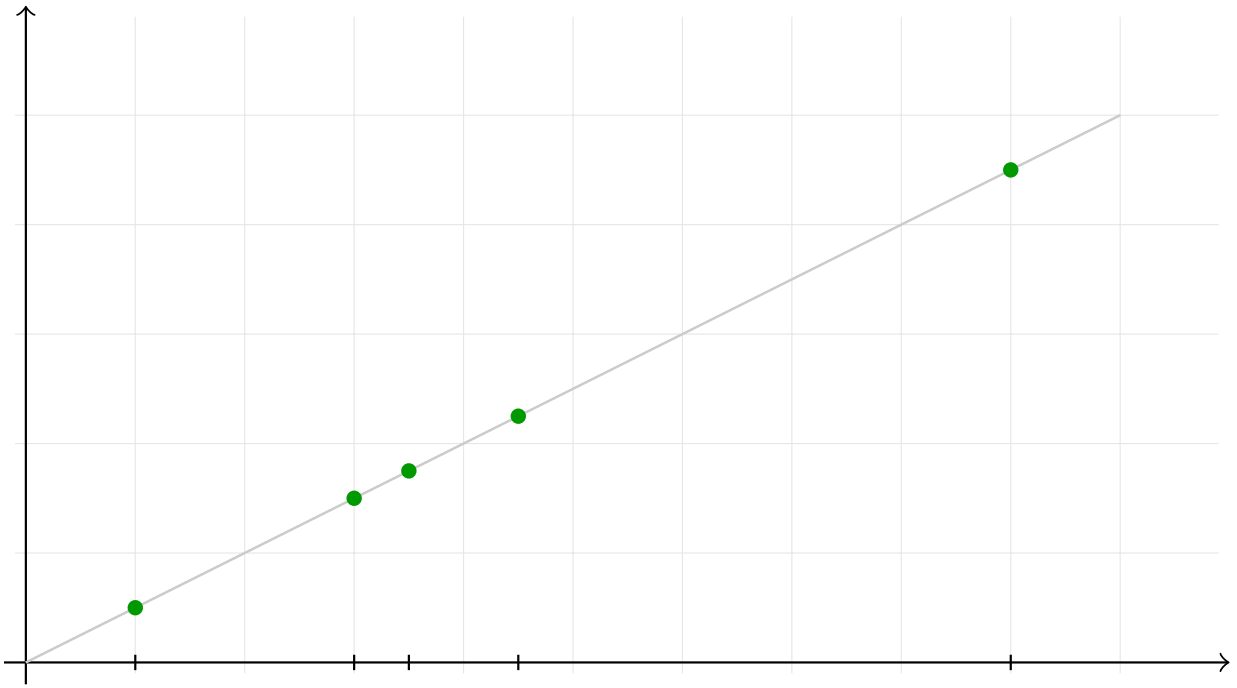
Making many linear combinations, we get either a many colinear points.

Encoding



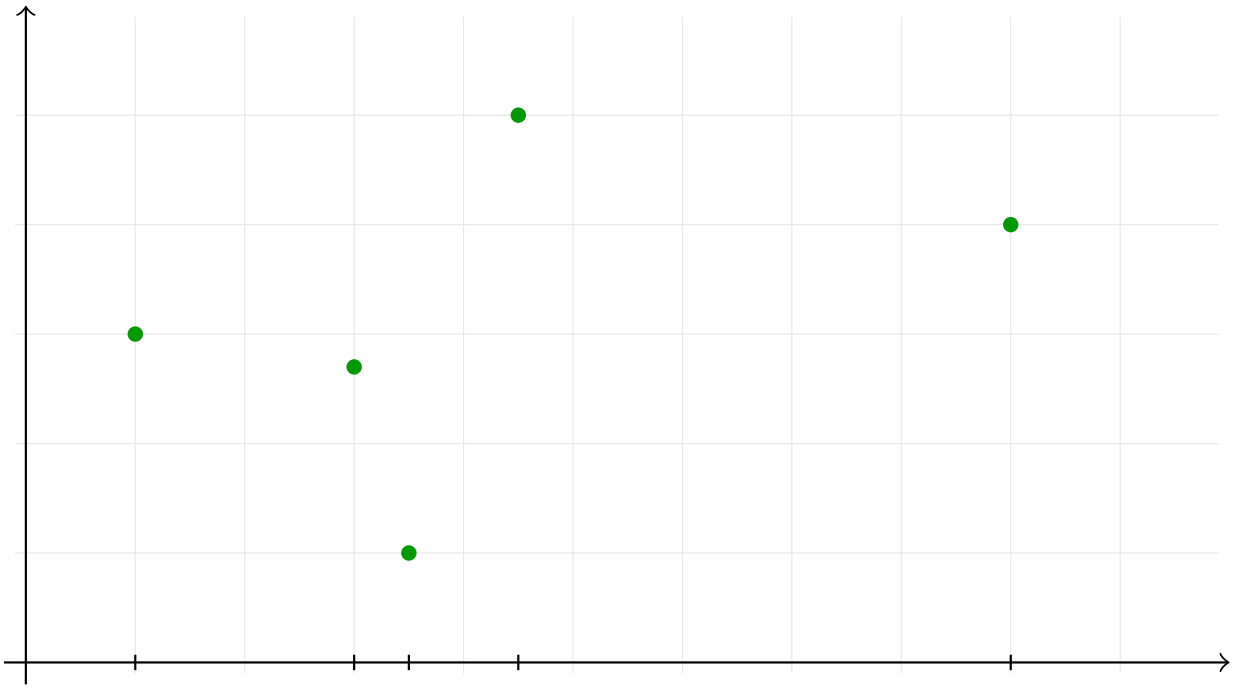
Making many linear combinations, we get either a many colinear points, or many random points.

Encoding



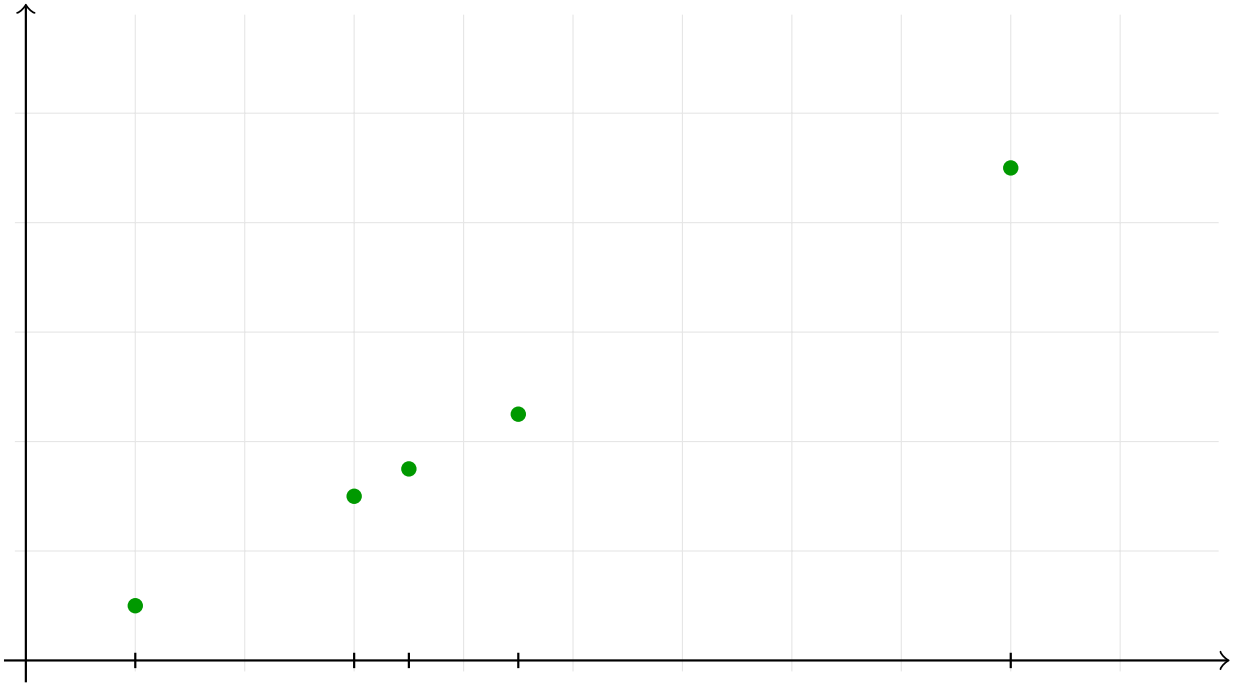
In other words, if $f : \{\text{option set}\} \to G$ is a random function, we cannot distinguish between $v \mapsto f(v)^s$ and a random function.

Encoding



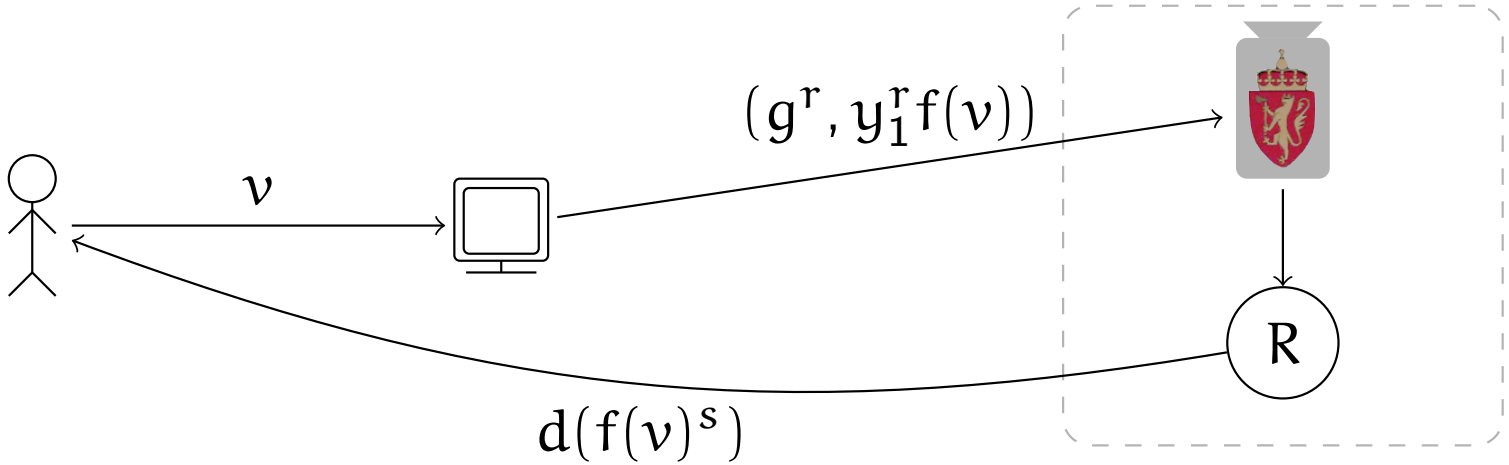
In other words, if $f : \{\text{option set}\} \to G$ is a random function, we cannot distinguish between $v \mapsto f(v)^s$ and a random function.

Encoding



But f will map options to small primes. Does $f(v)^s$ still look random?

Computing the Receipt Code



Full protocol: (i) Every player proves correctness of everything. (ii) An auditor supervises everything. (iii) A “verifiable shuffle” hides ballots from the auditor.

Threat Model & Security Goal

Security goal:

Ensure privacy of ballots case via honest computers and integrity of any ballot.

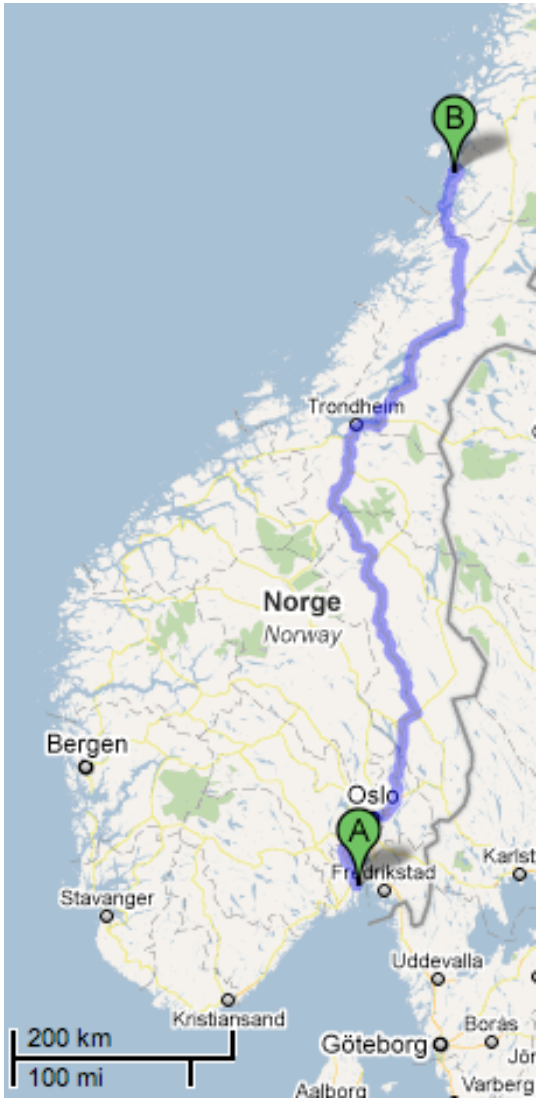
Threat model:

1. Axiomatic: Attacker controls the network.
2. Voters and computers can be corrupt.
3. At most one of the infrastructure players can be corrupt.
4. If the receipt generator is corrupt, no voters or computers can be corrupt.

Justification of Corruption Model

A The Receipt Generator

B The Ballot Box



Are we done?

Are we done?

No.

Question: Will it work?

Example: Denial-of-service I

If the attacker controls the network, he can target a denial-of-service attack geographically or perhaps even towards households.

Example: Denial-of-service II

Send SMS messages to voters saying that they have just voted.

Example: Implementation Flaw

If the voting software downloads ballot descriptions on-demand, the size of the download may reveal the voter choice. This possibility does not exist in the model the proof uses.

What else is missing from the model?

Example: Social Engineering Attack

The goal is to discard a vote.

A fake web site accepts the vote, then does nothing. A variant asks the voter for the correct receipt codes, then sends them to the voter via SMS (using the voter's Skype account).

In case of attack, do . . .