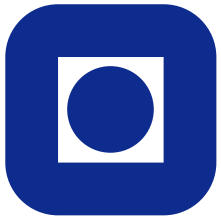


NTNU
Norwegian University of
Science and Technology

Electronic elections – National efforts

Kristian Gjøsteen

Dept. of Mathematical Sciences, NTNU

NISNet – Finse, April 26-30, 2010

Contents

Thursday

Electronic elections in Norway and how a cryptographic protocol was selected.

Friday

Analysing the cryptographic protocol.

Today's Contents

Norwegian elections

The E-valg 2011 Project

The Tender

Electronic ID in Norway

Norwegian Elections

The Norwegian voting procedure:

1. Select a party list. This is your ballot.
2. Modify the ballot (optional).

   Municipal Give personal votes to any candidate, from the chosen party or other parties. Order does not matter.

   County Give personal votes to candidate on the list. Order does not matter.

   Parliamentary Reorder or strike candidates on the list.

Norwegian Elections

The Norwegian voting procedure:

1. Select a party list. This is your ballot.
2. Modify the ballot (optional).

   Municipal Give personal votes to any candidate, from the chosen party or other parties. Order does not matter.

   County Give personal votes to candidate on the list. Order does not matter.

   Parliamentary Reorder or strike candidates on the list.

Note I

Write-in candidates effectively splits the vote between parties.

Norwegian Elections

The Norwegian voting procedure:

1. Select a party list. This is your ballot.
2. Modify the ballot (optional).

   Municipal Give personal votes to any candidate, from the chosen party or other parties. Order does not matter.

   County Give personal votes to candidate on the list. Order does not matter.

   Parliamentary Reorder or strike candidates on the list.

Note II

Ballots are essentially a sequence of *options* of length 15-100. The number of options is 80-1000.

Norwegian Elections

The Norwegian voting procedure:

1. Select a party list. This is your ballot.
2. Modify the ballot (optional).

   Municipal Give personal votes to any candidate, from the chosen party or other parties. Order does not matter.

   County Give personal votes to candidate on the list. Order does not matter.

   Parliamentary Reorder or strike candidates on the list.

Note III

Many ballots can be uniquely marked without influencing the result.

Norwegian Elections

The Norwegian voting procedure:

1. Select a party list. This is your ballot.
2. Modify the ballot (optional).

   Municipal Give personal votes to any candidate, from the chosen party or other parties. Order does not matter.

   County Give personal votes to candidate on the list. Order does not matter.

   Parliamentary Reorder or strike candidates on the list.

Electronic voting should not change voting patterns.

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. [...]*

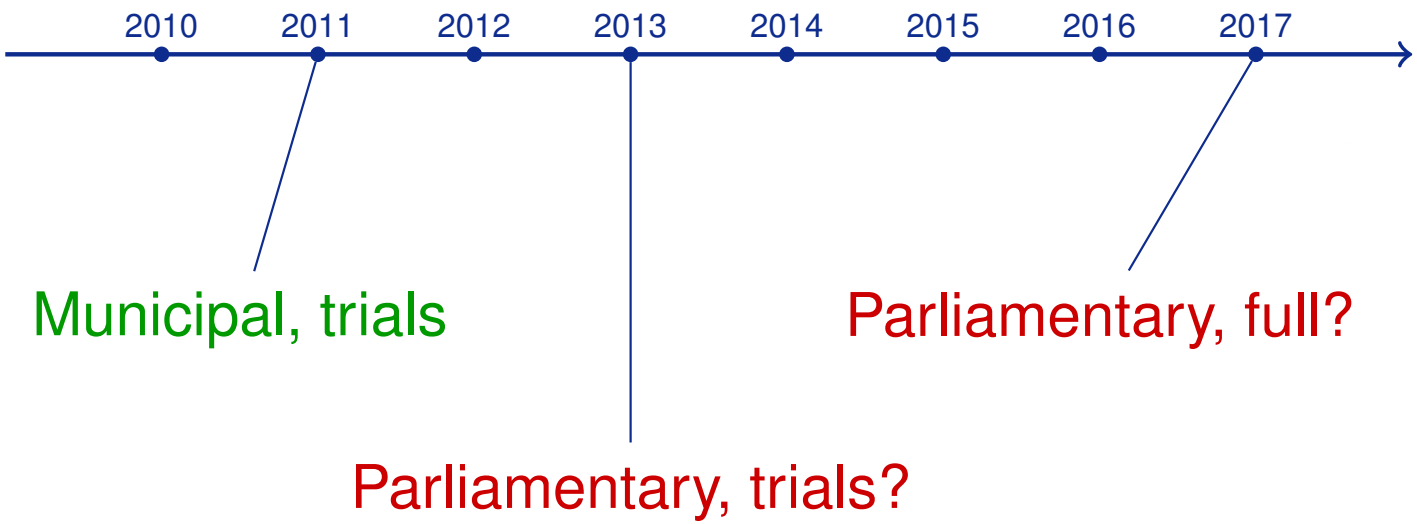
Note

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

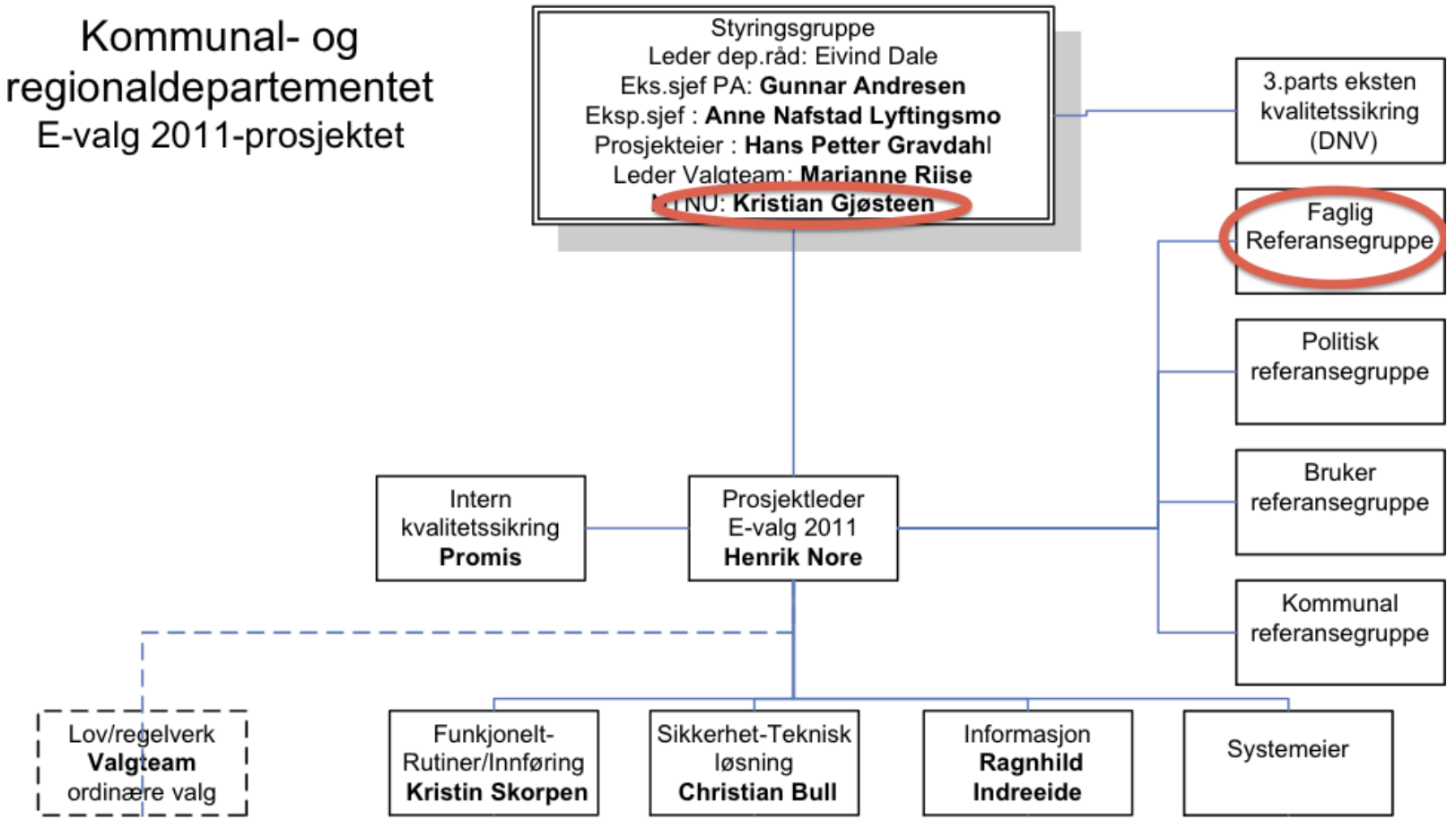
Note II

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.

Openness

Openness is a basic principle for the project:

1. Knowing the election mechanics is a democratic right.

Openness

Openness is a basic principle for the project:

1. Knowing the election mechanics is a democratic right.
2. Hypothesis: Openness implies more analysis, which implies fewer errors, which implies improved security.

Openness

Openness is a basic principle for the project:

1. Knowing the election mechanics is a democratic right.
2. Hypothesis: Openness implies more analysis, which implies fewer errors, which implies improved security.

— The three final bids are now public (except two documents).

Openness

Openness is a basic principle for the project:

1. Knowing the election mechanics is a democratic right.
2. Hypothesis: Openness implies more analysis, which implies fewer errors, which implies improved security.

— The three final bids are now public (except two documents).

— The source code will be public (but not «open source»).

Openness

Openness is a basic principle for the project:

1. Knowing the election mechanics is a democratic right.
2. Hypothesis: Openness implies more analysis, which implies fewer errors, which implies improved security.

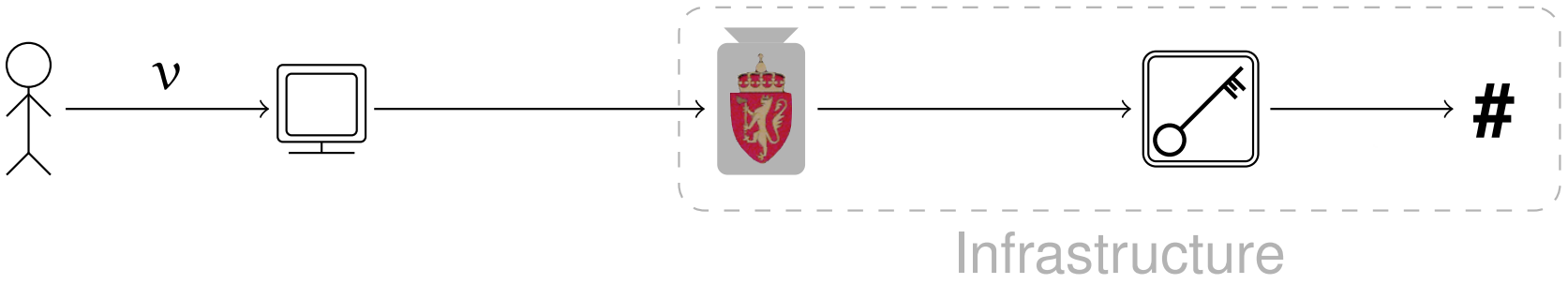
— The three final bids are now public (except two documents).

— The source code will be public (but not «open source»).

Note

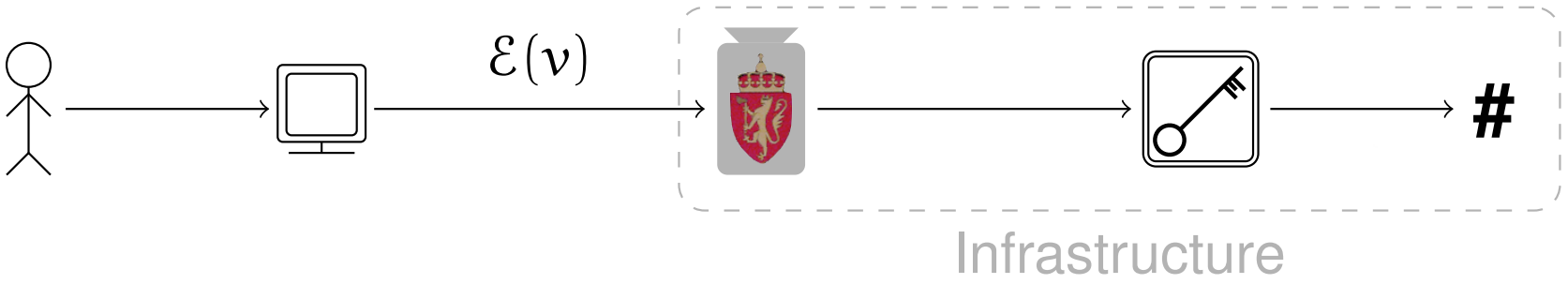
Universal verifiability is hard.

Internet Elections: Basic Idea



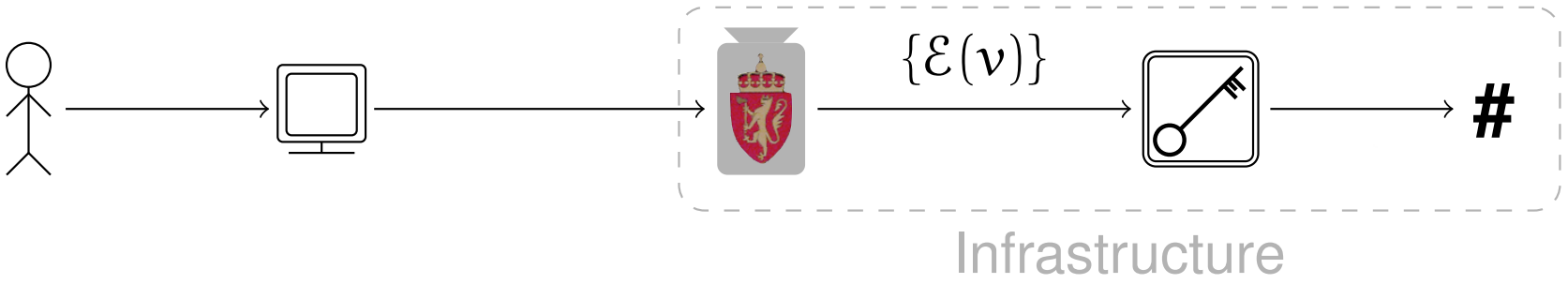
— The voter tells his pc what to vote.

Internet Elections: Basic Idea



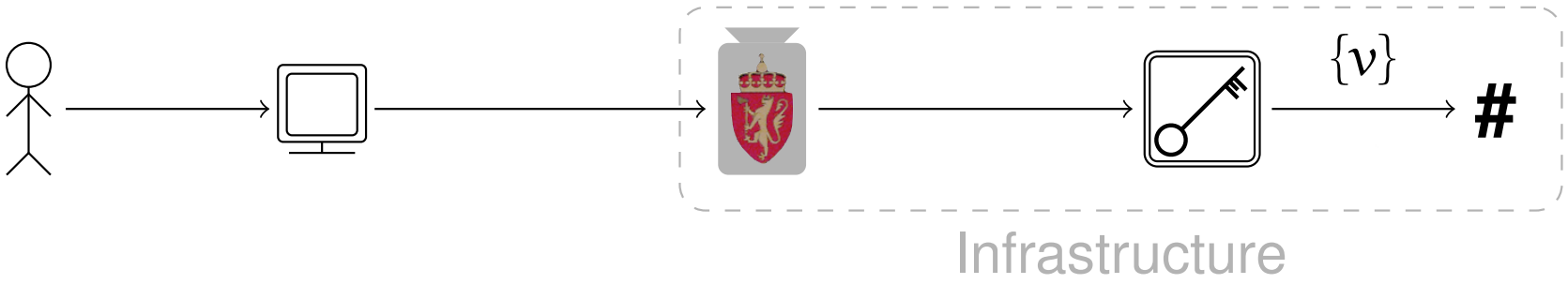
— The pc encrypts the vote and submits it to a ballot box.

Internet Elections: Basic Idea



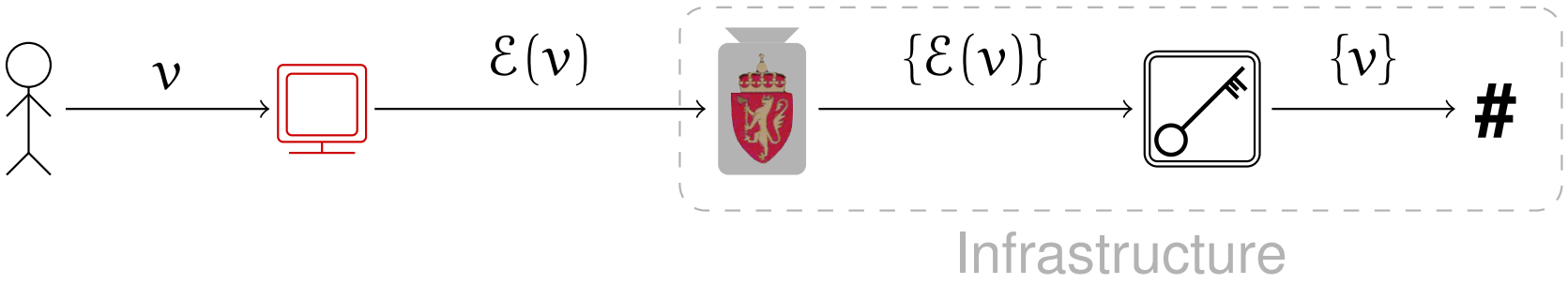
— After voting is done, the ballot box shuffles ciphertexts and submits them for decryption.

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.

Political Requirement: Electronic ID

For some time, the Norwegian goverment has been trying to establish an identity portal, a single gateway for identification that can be used by government web sites.

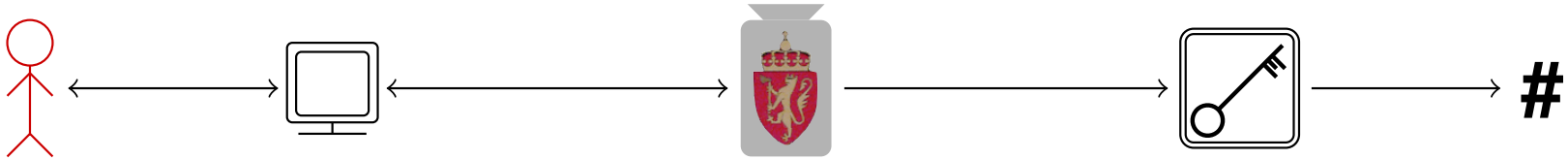
Political Requirement: Electronic ID

For some time, the Norwegian goverment has been trying to establish an identity portal, a single gateway for identification that can be used by government web sites.

The internet voting solution must use the identity portal.

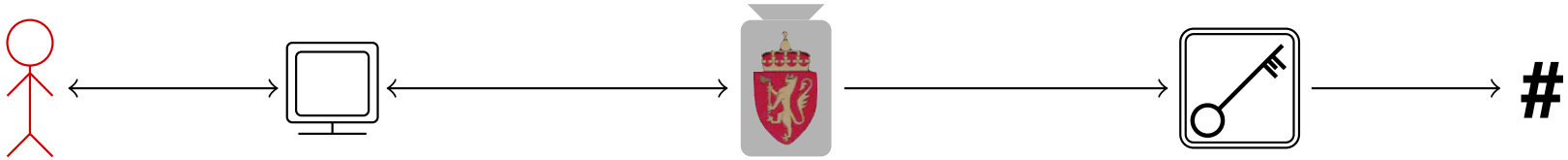
More on this later.

Functional Requirement: Usability



Our voter finds himself participating in a cryptographic protocol.

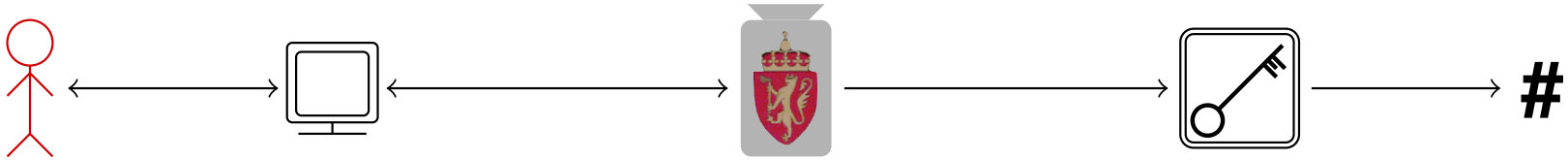
Functional Requirement: Usability



Our voter finds himself participating in a cryptographic protocol.

— Voters won't even do lightweight crypto.

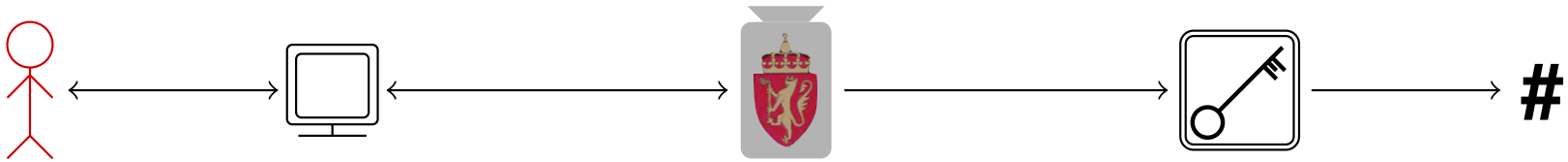
Functional Requirement: Usability



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



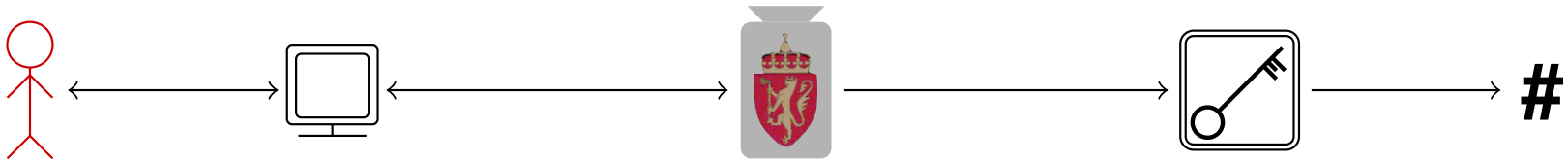
Our voter finds himself participating in a cryptographic protocol.

- 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



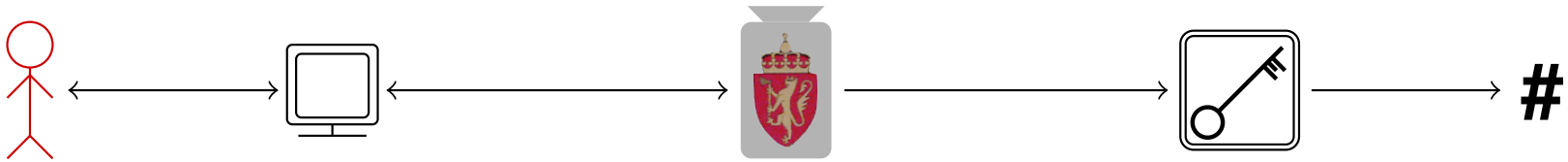
Our voter finds himself participating in a cryptographic protocol.

- 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.

Functional Requirement: Speed

The ballot box will be open for several weeks.

- Most votes will be submitted over six hours on the final day.
- Avoid last-minute queuing by closing at four am.

Up to 100 000 votes will be submitted during six hours for the trial, 2 million for full-scale deployment.

Functional Requirement: Speed

The ballot box will be open for several weeks.

- Most votes will be submitted over six hours on the final day.
- Avoid last-minute queuing by closing at four am.

Up to 100 000 votes will be submitted during six hours for the trial, 2 million for full-scale deployment.

Example

The protocol requires 500-1600 infrastructure exponentiations per ballot for *active security*, which is expensive. Other proposals required much more.

Functional Requirement: Speed

The ballot box will be open for several weeks.

- Most votes will be submitted over six hours on the final day.
- Avoid last-minute queuing by closing at four am.

Up to 100 000 votes will be submitted during six hours for the trial, 2 million for full-scale deployment.

Decrypting 2 million ballots should require at most 30 minutes.

Security Requirement: Privacy

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

Security Requirement: Integrity

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

Security Requirement: Integrity

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

Note

Every option matters.

Security Requirement: Buying Votes

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

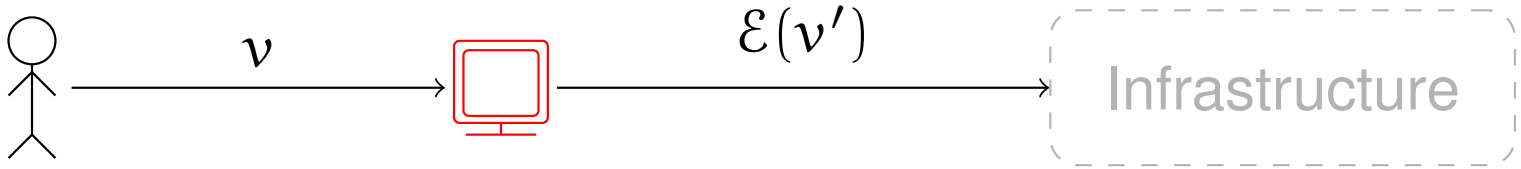
Security Requirement: Buying Votes

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

Note

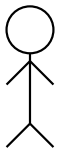
Universal verifiability is hard!

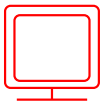
Problem: Compromised Computers



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

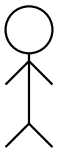
Problem: Compromised Computers

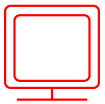




— Compromised computers were considered by Chaum's *SureVote* (2000) and UK's CESG's *e-Voting Security Study* (2002).

Problem: Compromised Computers

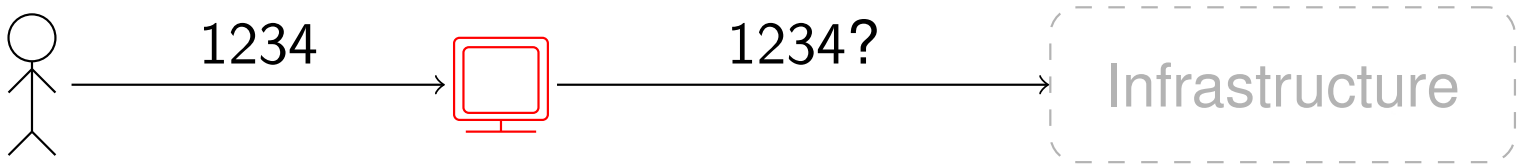




— Every personal voting card has a unique table:

Candidate	Vote Code	Receipt Code
A	1234	5678
B	4321	0987

Problem: Compromised Computers

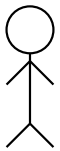


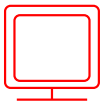
— Every personal voting card has a unique table:

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



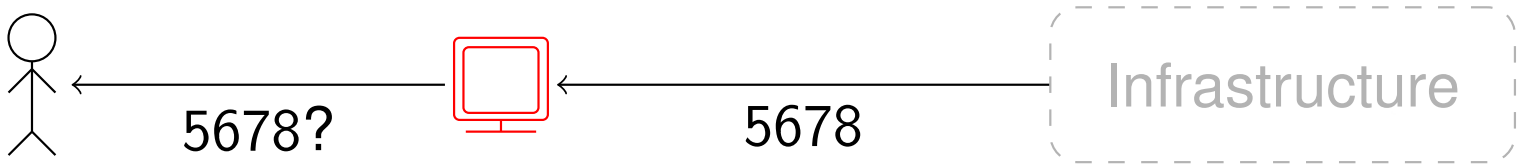


— Every personal voting card has a unique table:

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

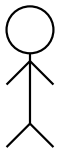


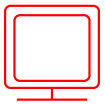
— Every personal voting card has a unique table:

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

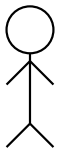


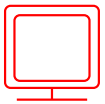


— Compromised computers were considered by Chaum's *SureVote* (2000) and UK's CESG's *e-Voting Security Study* (2002).

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.

Tender: Competitive Dialogue

> *In the first phase [. . .], the goal is to identify and define the optimum method to meet the specified requirements. All aspects of the contract can be discussed in the dialogue, [...]*
>
> *When the invitation to submit a final tender is sent out, the dialogue closes.*

Essentially, the requirements specification is written with input from the bidders.

The Requirements Specification

OS4.1 The election system shall ensure the secrecy of the any vote at all stages of an election.

OS8.1 The election system shall ensure that E-votes, that have been acknowledged, will not be lost or altered under any circumstance.

OS8.10 The election system shall supply the e-voter with a message whereby the voter can ascertain that the vote was recorded as intended. The message may be either out-of band or protected against outside manipulation. The message may contain one or more shared secrets provided to the voter prior to the e-voting phase of the election. [. . .]

The Requirements Specification

MC2 The exact licensing scheme is not particularly important, and the supplier may utilize any suitable, established open source licensing scheme which enables the supplier to conform to the requirements of the contract. However the license must allow the Ministry to publish the source code, and it must be possible for anyone to examine and compile the source code.

MC3 The license must allow the Ministry or anyone the Ministry authorizes to modify the code for use in academic research or in further development of the system for use in Norwegian elections.

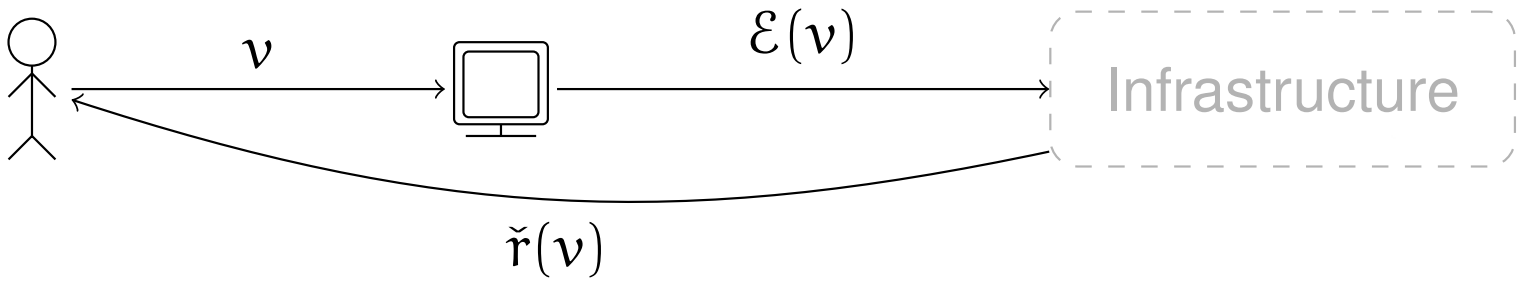
Tender: Analysis

Academic groups from UiB and NTNU helped with security analysis during the tender and after the final bids.

- Preliminary proposals were subjected to mild analysis (and strong criticism).
- The three final bids received further analysis (and some mild criticism).

In addition, there was internal analysis. Other external parties contributed, among them NSM.

The Final Bids



— Every vendor eventually adapted the SureVote-type system by dropping the Vote Code and keeping the Receipt Code.

— They use mixing before decryption.

— Technical solutions were very different.

Evaluation Results

Evaluation Criteria		Decided Weight %	Computas		ErgoGroup		Indra	
1. Cost (20-30 %)		**30 %**	**Cost**	**Points (0-10)**	**Cost**	**Points (0-10)**	**Cost**	**Points (0-10)**
1.1	Total cost	30 %	NOK 107 012 624	**5,3**	NOK 56 614 741	**10,0**	NOK 105 663 877	**5,4**
2. Tenders competence, implem		**30 %**	**Score**	**Points**	**Score (0-3)**	**Points**	**Score (0-3)**	**Points**
2.1	Competence	10 %	2,8	0,3	2,7	0,3	2,4	0,2
2.2	Implementation Methodology	10 %	2,1	0,2	2,0	0,2	1,9	0,2
2.3	References	10 %	2,6	0,3	2,6	0,3	2,6	0,3
	SUM	**30 %**		**0,8**		**0,7**		**0,7**
	Normalized (0-10)			**10,0**		**9,7**		**9,1**
3. Proposed solution (30-40%)		**40 %**	**Score**	**Points**	**Score (0-3)**	**Points**	**Score (0-3)**	**Points**
3.1	General Requirements	1 %	1,8	0,0	2,1	0,0	1,7	0,0
3.2	Technical Requirements	5 %	2,0	0,1	2,0	0,1	2,0	0,1
3.3	Functional Requirements	13 %	2,0	0,3	2,0	0,3	1,9	0,3
3.4	Security	11 %	1,9	0,2	1,9	0,2	1,3	0,1
3.5	Accessibility and usability	10 %	1,2	0,1	1,3	0,1	1,3	0,1
	SUM	**40 %**		**0,7**		**0,7**		**0,6**
	Normalized (0-10)			**9,8**		**10,0**		**8,6**
	Overall SUM	**100 %**		**8,5**		**9,9**		**7,8**

The Winner

The winning bid was submitted by ErgoGroup and Scytl.

- Ergo does election administration and paper ballot counting.
- Scytl does internet voting.

I am happy about this outcome.

Changes to the Proposal

After selecting the winning bid, the E-valg 2011 project asked for some changes to the winning proposal:

- The exact details of how receipt codes are generated.
- Out-of-band communication with the voter.
- Better security against active attacks by the infrastructure.
- A better specification of auditing.

Changes to the Proposal

After selecting the winning bid, the E-valg 2011 project asked for some changes to the winning proposal:

- The exact details of how receipt codes are generated.
- Out-of-band communication with the voter.
- Better security against active attacks by the infrastructure.
- A better specification of auditing.

Why?

To make it easier to prove things about the protocol.

Electronic ID in Norway

Plans for electronic identities in Norway have revolved around plans for an identity portal for all government web sites. The original idea was to buy identity services from commercial providers.

Identity 1.0

The government believes identity = name.

Electronic ID in Norway

Plans for electronic identities in Norway have revolved around plans for an identity portal for all government web sites. The original idea was to buy identity services from commercial providers.

Identity 1.0

The government believes identity = name.

Chicken and egg problem

If nobody has an electronic identity, no web site will support electronic identities. But if no web site supports electronic identities, nobody will acquire electronic identities.

Electronic ID in Norway

Plans for electronic identities in Norway have revolved around plans for an identity portal for all government web sites. The original idea was to buy identity services from commercial providers.

Identity 1.0

The government believes identity = name.

Business model problem

Who should pay?

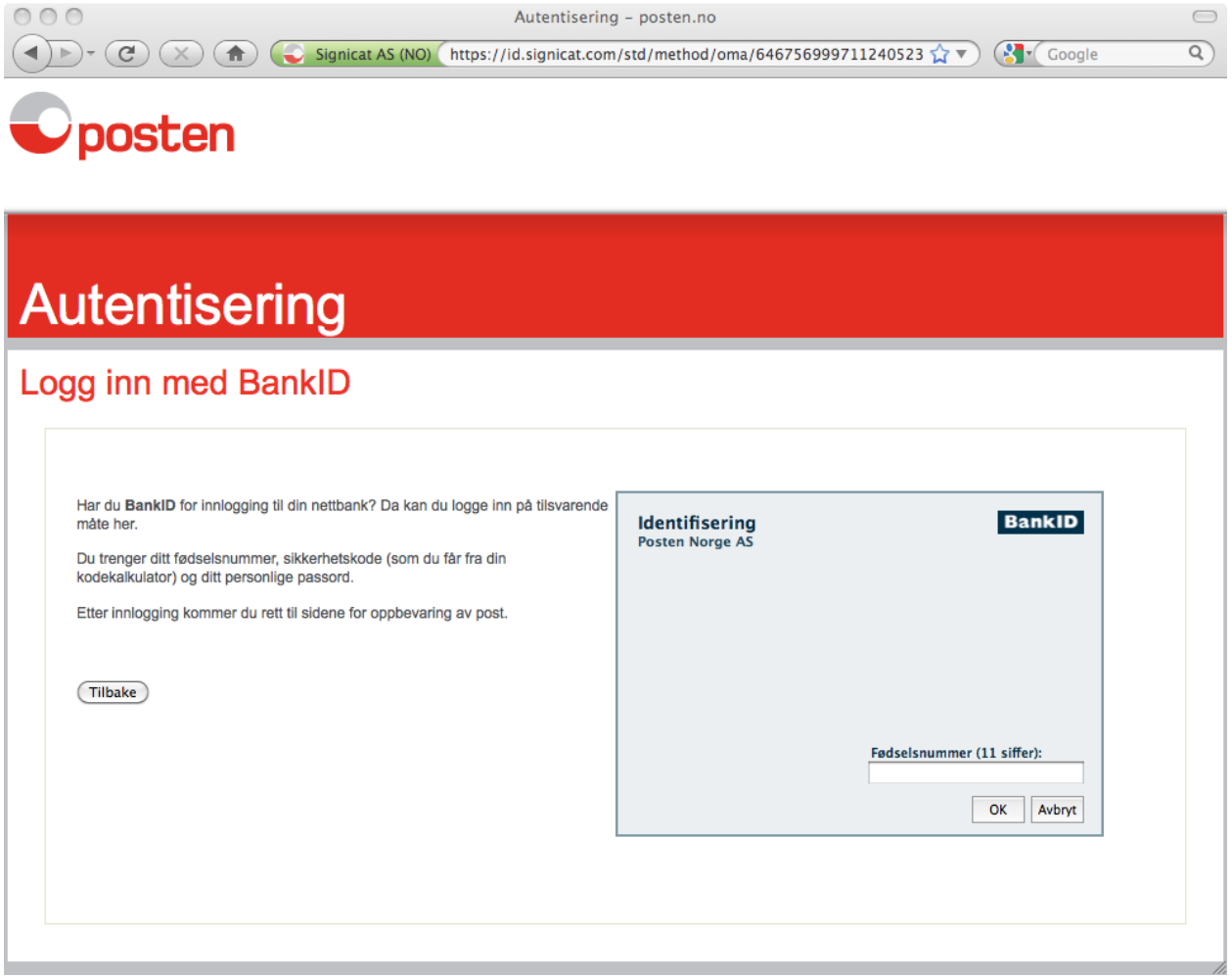
Commercial: BankID

- Banks already provide identity cards.
- Internet banks already have a (secure!) login mechanism.
- Everyone does internet banking.

Business case

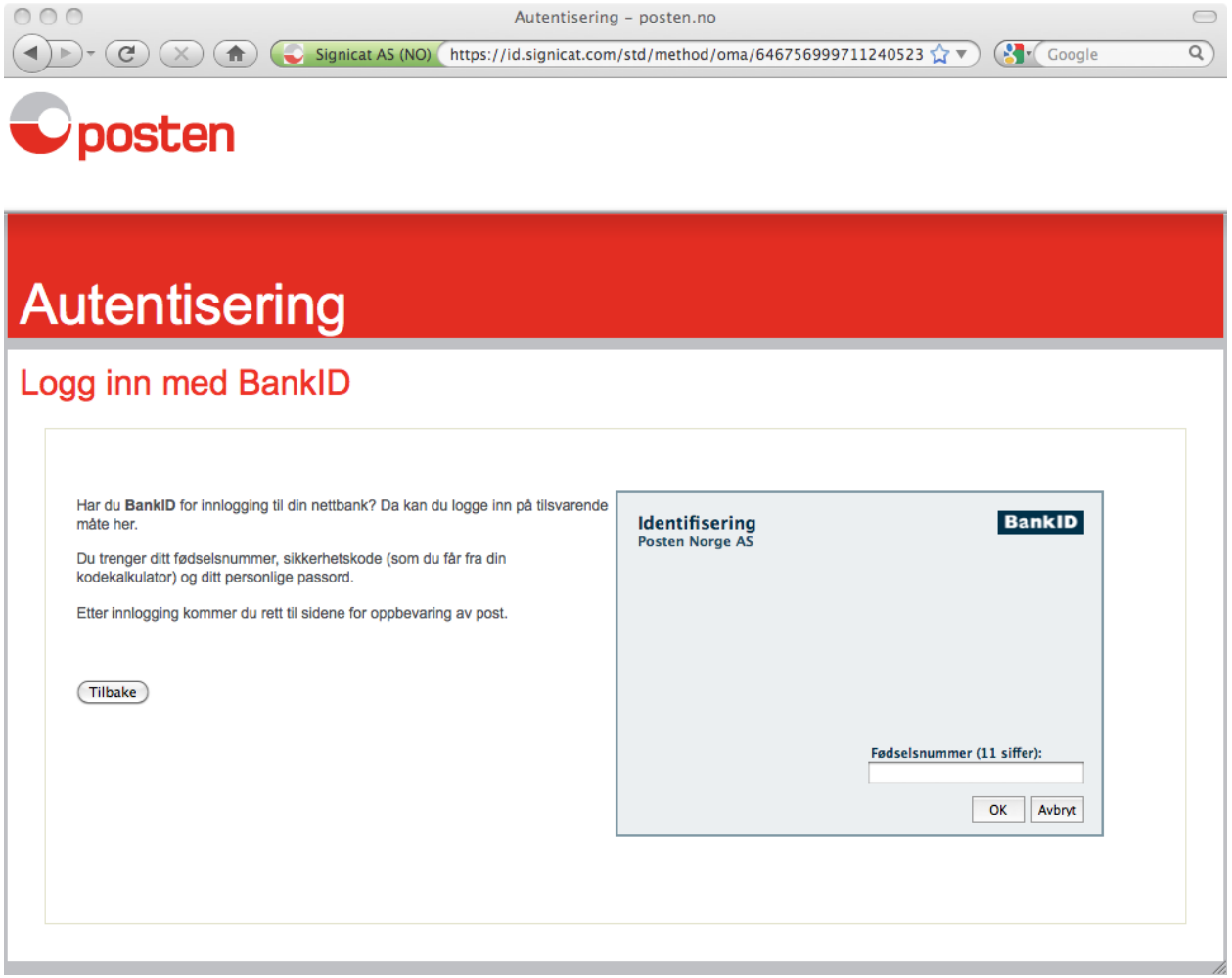
Reuse internet banking login at other web sites. Charge the web sites!

Commercial: BankID



Embedded Java applet accepts password and pin code.

Commercial: BankID



A phisherman's dream.

Commercial: BankID

Two problems:

- The banks control a user's signing keys: Could banks vote on behalf of voters?
- Privacy?

Commercial: BankID

Internet banking origins implies weakness today:

- Couples share passwords and pin-codes for their internet bank. This is sound and secure.
- Internet bank login changes to an identity mechanism.
- Couples still share passwords and pin-codes. This is unsound and possibly insecure.

User behaviour hasn't changed, but system changes has made previously secure behaviour in secure.

Commercial: BankID

Details about the system are secret, but there is significant evidence of too weak quality control:

— Y. Espelid, L.-H. Netland, A. N. Klingsheim, and K. J. Hole, *Robbing Banks with Their Own software—an Exploit against Norwegian Online Banks*, in Proc. 23rd International Information Security Conference (SEC 2008), Milan, Italy, September 8-10, 2008.

— K. Gjøsteen, *Weaknesses in BankID, a Norwegian PKI substitute deployed by Norwegian banks*. Proceedings of EuroPKI 2008, volume 5057 of LNCS, pages 196-206.

We have no positive reason to trust this system.

Commercial: BankID

Recent development: BankID will not be used in the governments identity portal, *because BankID does not provide message encryption.*

Commercial: Buypass

— National gambling monopoly has distributed smart cards to gamblers (nearly everyone!).

Business case: Add proper certificates to smart cards. Charge web sites!

Commercial: Buypass

- This is a smartcard-based system, but secret.
- It uses a Java plugin. It is not integrated with web browsers or operating systems.
- Privacy?

We have no positive reason to trust this system.

Commercial: Buypass

But:

- Most deployed smartcards are camouflaged as gambling cards.
- Users acquired them as gambling cards.
- Users probably treat them as just gambling cards.
- It is unknown how many share card and pin-codes with partners, children, neighbours, etc.

We don’t know if this system is sufficiently secure in practice.

Commercial: Commfides

Not marketed to consumers.

Goverment: Identity Card

The government plans a national identity card that will also contain an electronic identity.

- Delayed.
- Secret. But probably a standard smartcard based system.

Goverment: Identity Card

The government plans a national identity card that will also contain an electronic identity.

— Delayed.

— Secret. But probably a standard smartcard based system.

Hope

The government has recently hired cryptographers to work on this: there is hope for security and an enlightened approach to secrecy.

Goverment: Identity Card

The government plans a national identity card that will also contain an electronic identity.

- Delayed.
- Secret. But probably a standard smartcard based system.

Hope

The government has recently hired cryptographers to work on this: there is hope for security and an enlightened approach to secrecy.

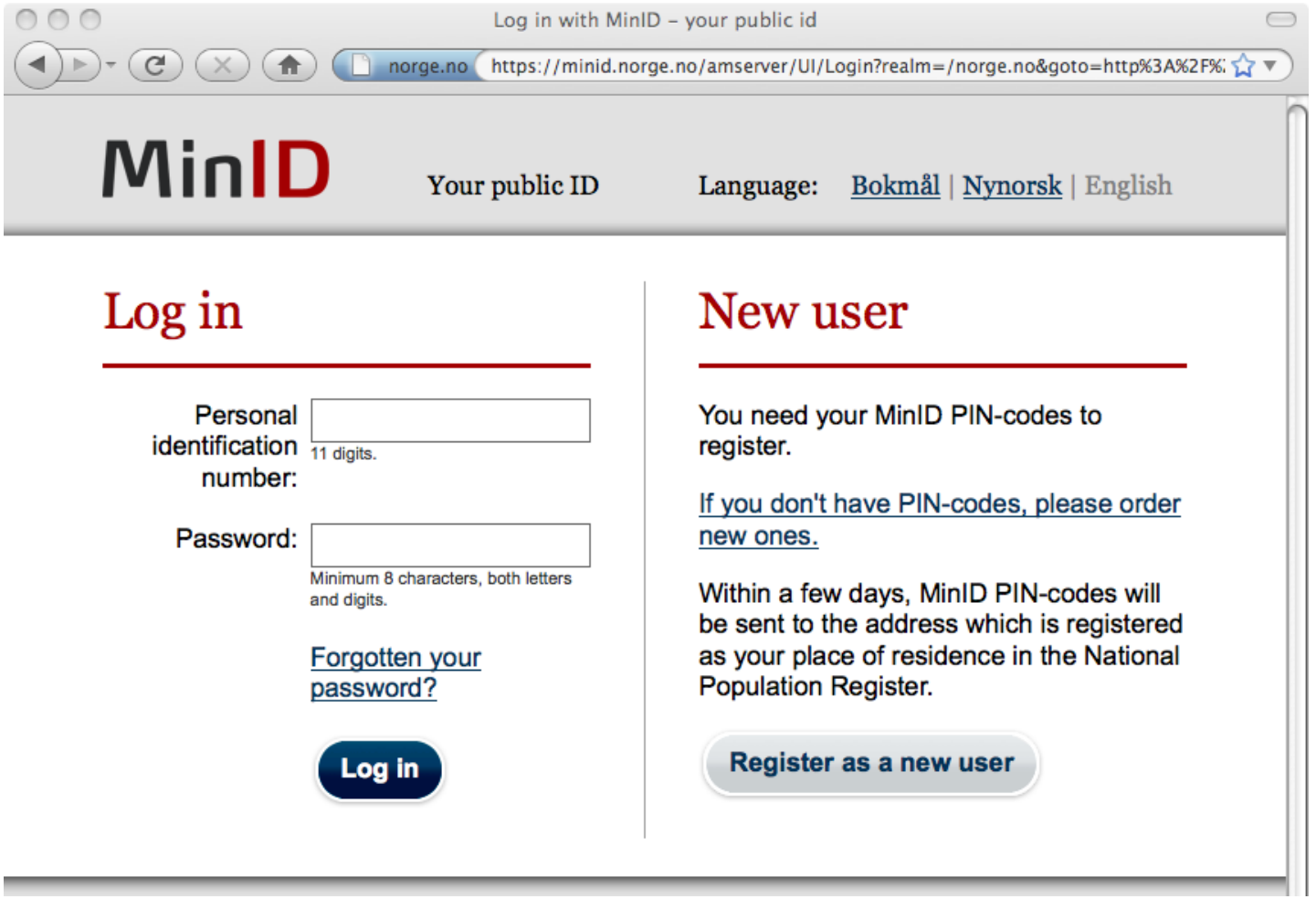
Chicken and egg problem

There is no killer application. Since the user must travel to a police station to provide a fingerprint, deployment will likely be slow.

Government: MinID

- A light-weight pin-based system was deployed for electronic submission of tax returns.
- This system has been extended to a light-weight single-sign-on system for government web sites.
- Password + pin from mobile phone.

Government: MinID



Government: MinID

- SAML-based federated single-sign-on system.
- People use this identity.
- It may be personal.
- Privacy?
- If you ask for security analysis, you get no response.

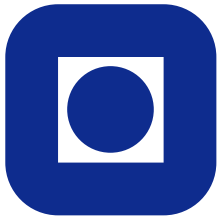
We have no positive reason to trust this system.

Government: MinID

MinID will be used.

- Signatures are not supported by MinID, but we can fake it.
- Federation should be turned off.
- Usability mess? Some codes from SMS messages should be typed in, some should *not*.

Buypass may also have to be used.



NTNU
Norwegian University of
Science and Technology

Electronic elections – National efforts

Kristian Gjøsteen

Dept. of Mathematical Sciences, NTNU

NISNet – Finse, April 26-30, 2010

Contents

Thursday

Electronic elections in Norway and how a cryptographic protocol was selected.

Friday

Analysing the cryptographic protocol.

Today's Contents

Proving Security

Core Protocol

Full Protocol

Threat Model & Security Goal

Security Proof

Are we done?

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.

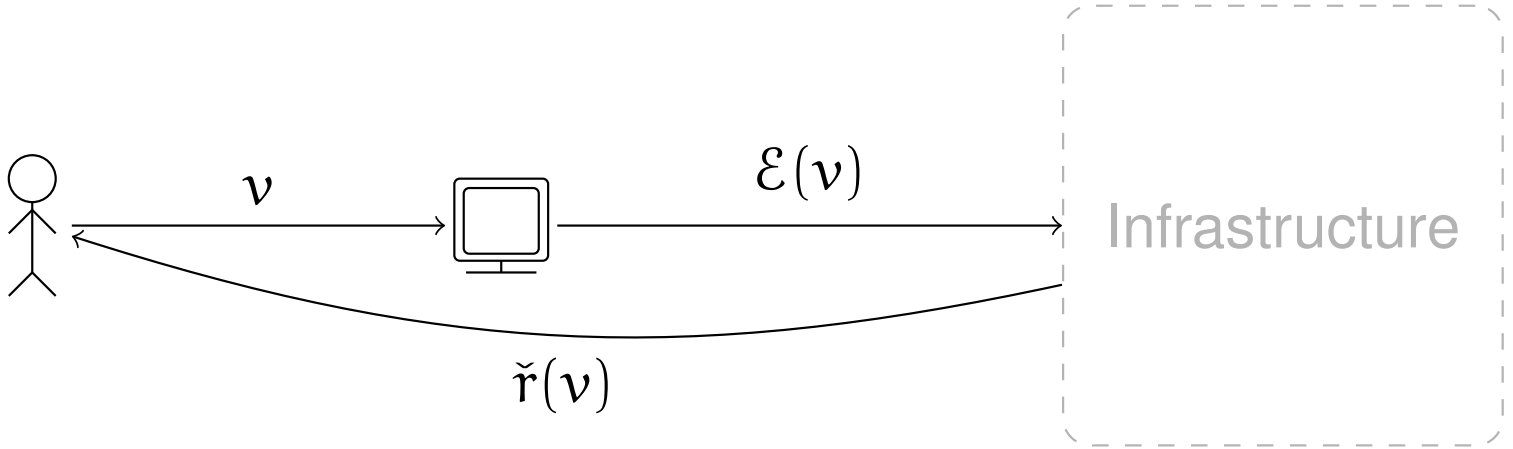
2. We usually prove that a certain class of attackers cannot exist. This class is usually characterized by a security goal and a threat model (attacker capability). We need to get the class right.
    - Example I: There was a long debate about the «correct» class of attackers for public key encryption.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

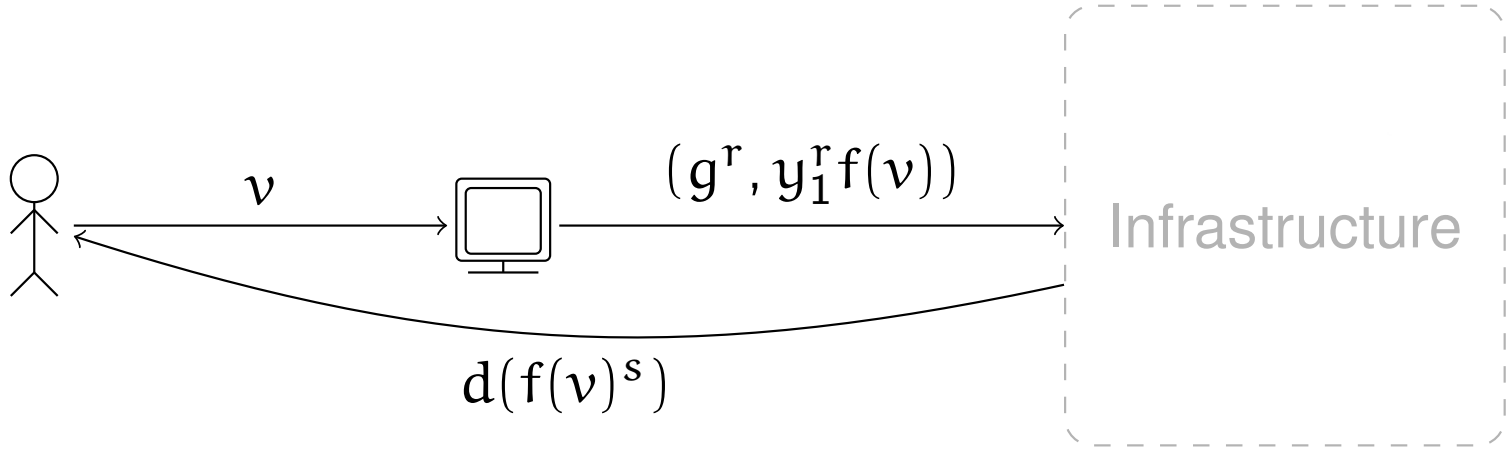
Provable security can be used to increase confidence in cryptographic protocols.

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.

Computing the Receipt Code

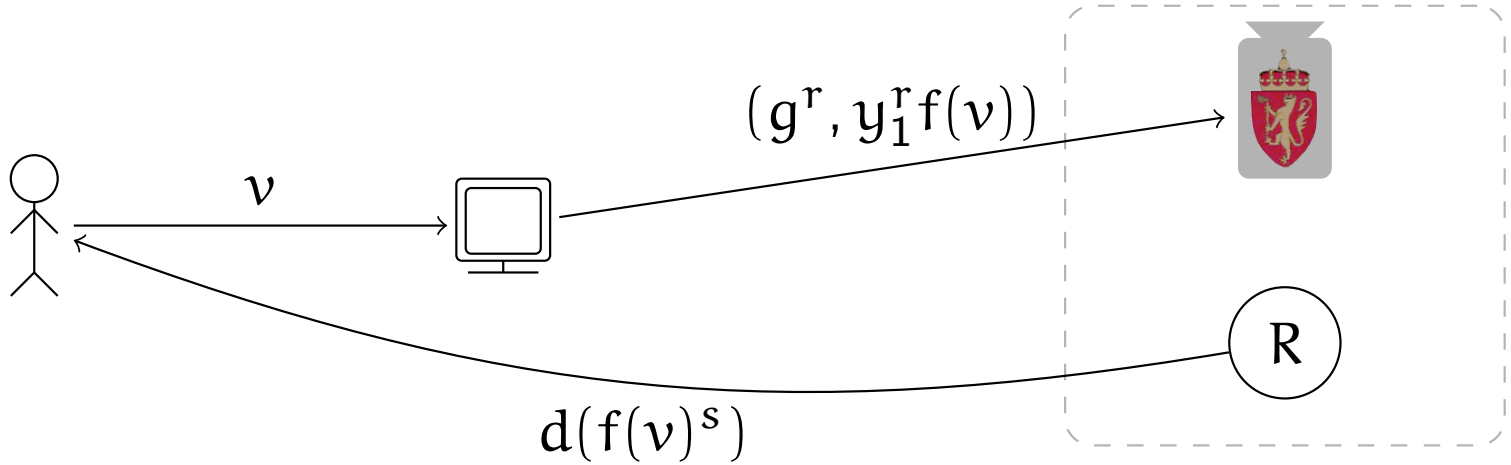


Computing the Receipt Code



- f $\{\text{option set}\} \to G$
- s per-voter exponent, random
- d $G \to \{\text{receipt code set}\}$, per-voter, pseudo-random
- y_1 election encryption key

Computing the Receipt Code



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

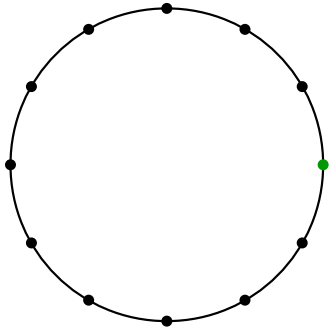
ElGamal, Geometry and Donuts

Here's a different perspective on ElGamal encryption.

Note

These donuts aren't related to elliptic curve donuts.

ElGamal, Geometry and Donuts

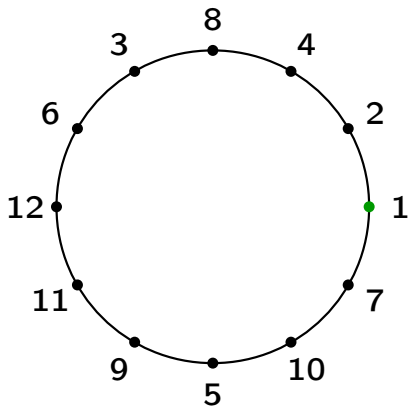


We think of a finite cyclic group $\mathbb{G}$ as a bunch of elements arranged on a circle such that the group operation corresponds to angle addition.

Note

Angle multiples corresponds to exponentiations in the group.

ElGamal, Geometry and Donuts

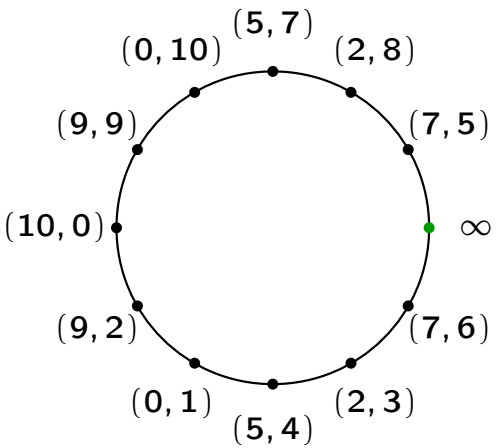


We think of a finite cyclic group G as a bunch of elements arranged on a circle such that the group operation corresponds to angle addition.

Note

Angle multiples corresponds to exponentiations in the group.

ElGamal, Geometry and Donuts

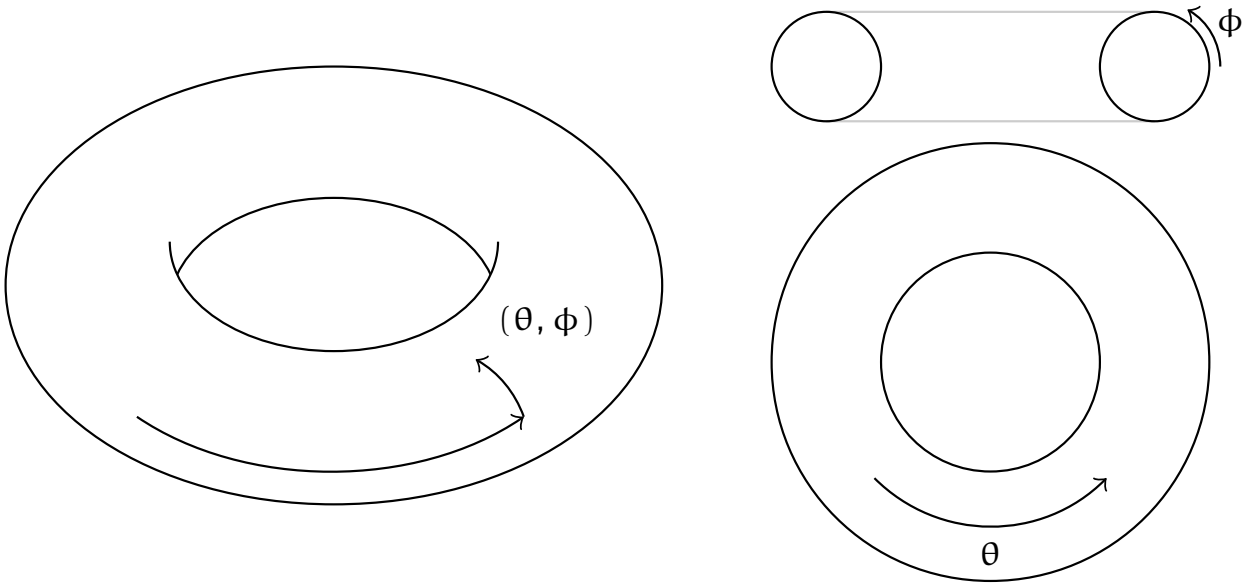


We think of a finite cyclic group G as a bunch of elements arranged on a circle such that the group operation corresponds to angle addition.

Note

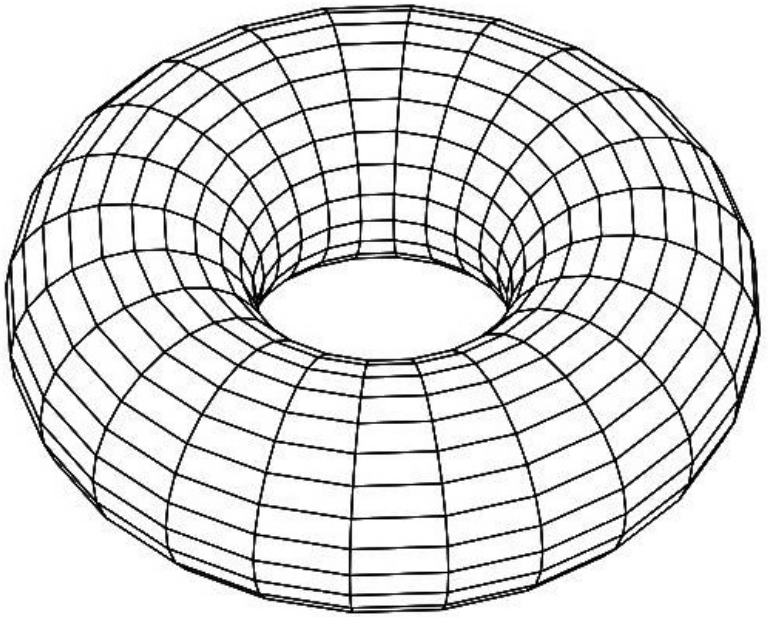
Angle multiples corresponds to exponentiations in the group.

ElGamal, Geometry and Donuts



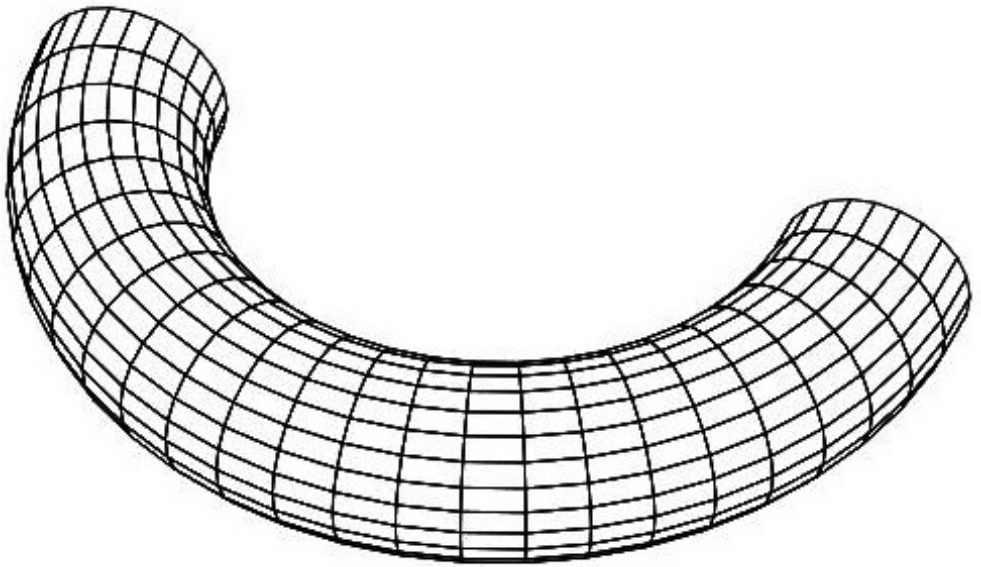
If a group element corresponds to an angle, a pair of group elements corresponds to a point of the surface of a torus.

ElGamal, Geometry and Donuts



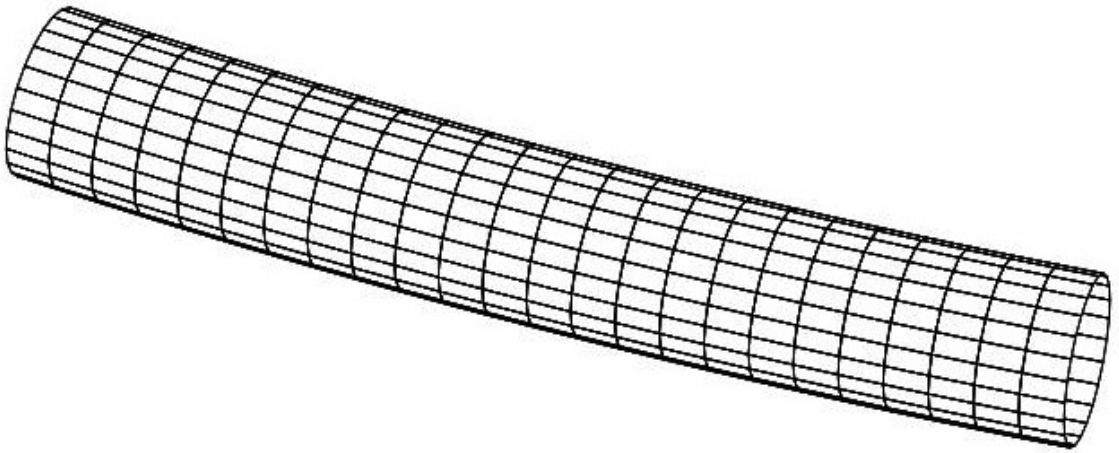
The surface of a torus is essentially a plane, just like the ordinary plane.

ElGamal, Geometry and Donuts



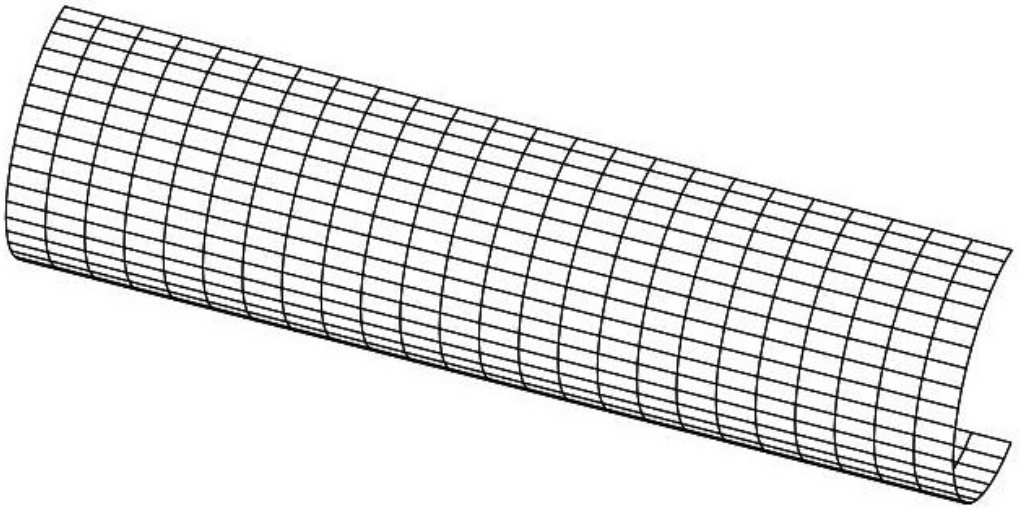
The surface of a torus is essentially a plane, just like the ordinary plane.

ElGamal, Geometry and Donuts



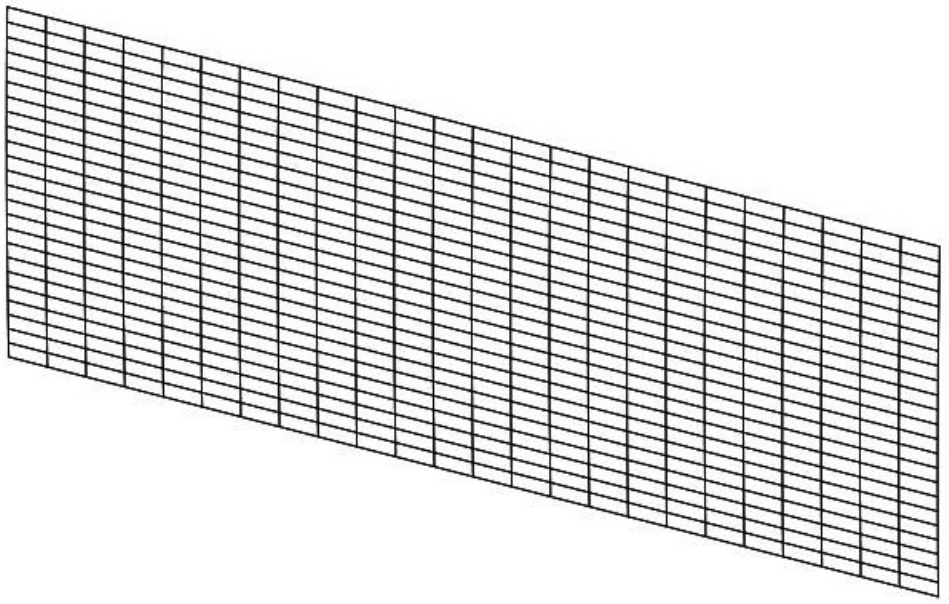
The surface of a torus is essentially a plane, just like the ordinary plane.

ElGamal, Geometry and Donuts



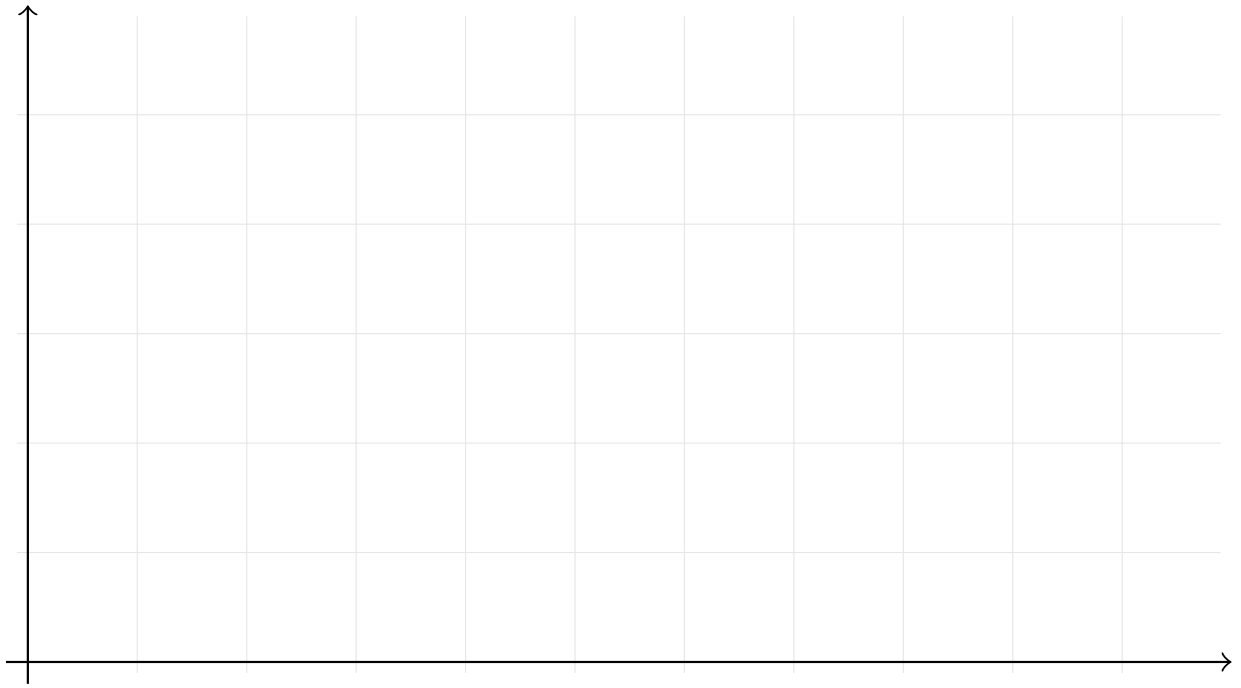
The surface of a torus is essentially a plane, just like the ordinary plane.

ElGamal, Geometry and Donuts



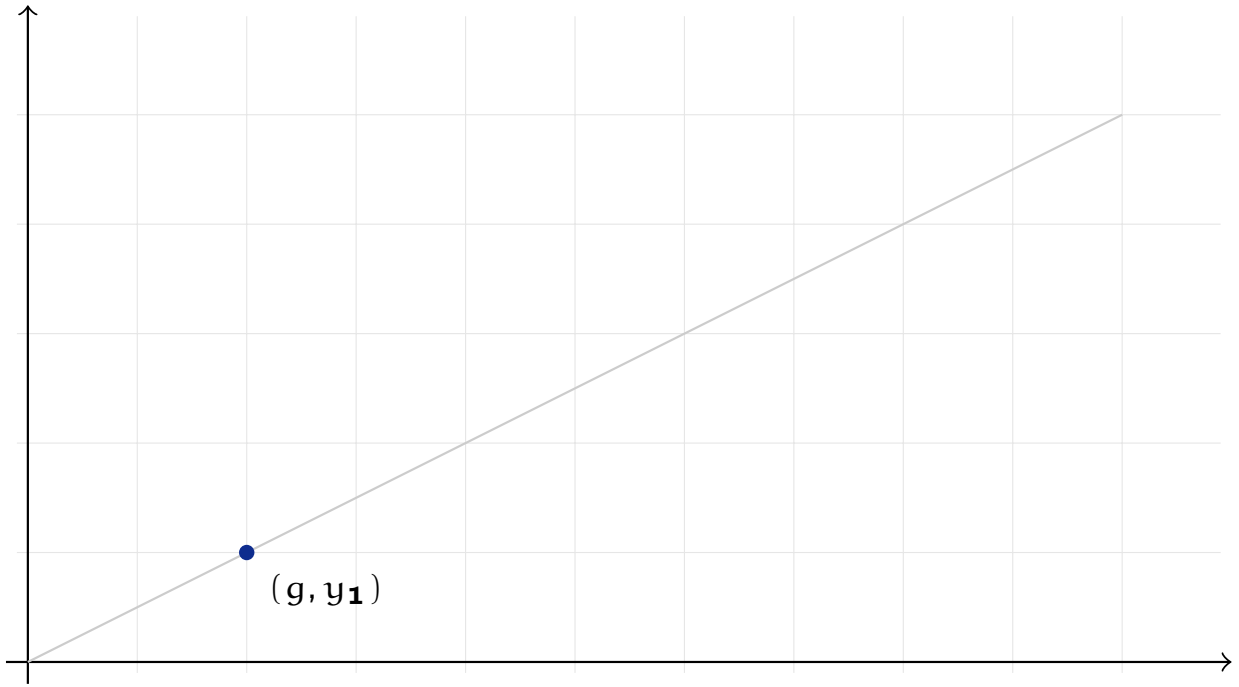
The surface of a torus is essentially a plane, just like the ordinary plane.

ElGamal, Geometry and Donuts



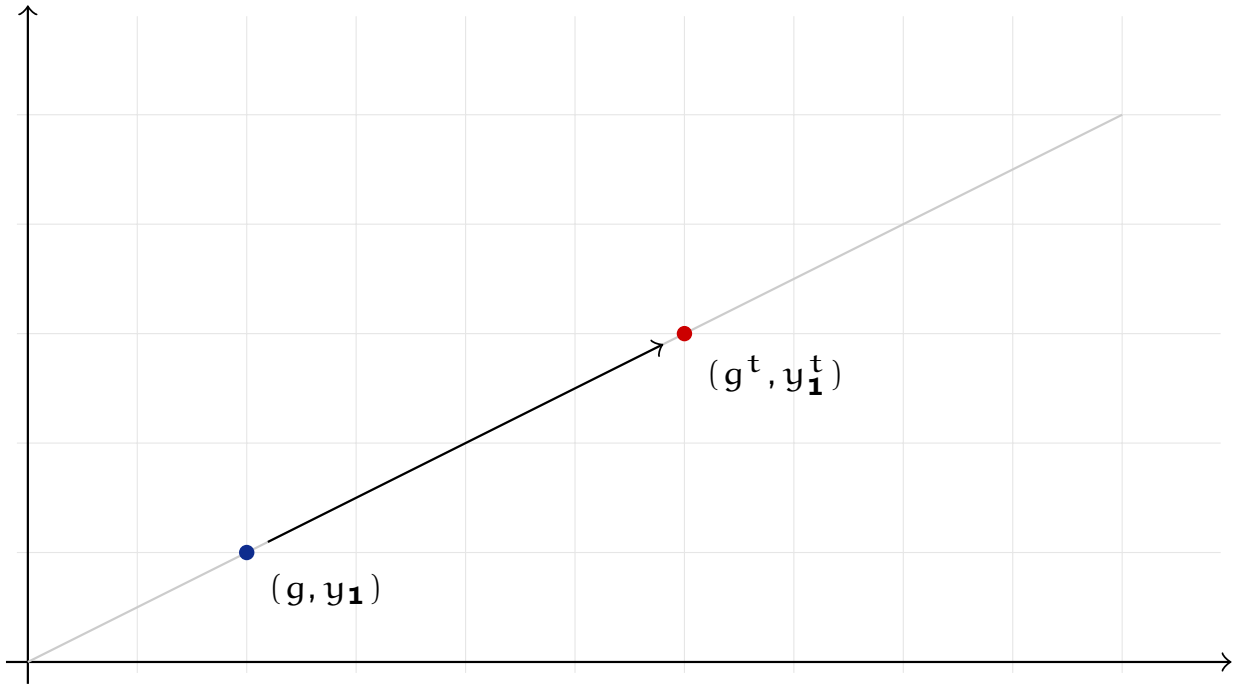
In other words: We think of pairs of group elements as points in an ordinary plane.

ElGamal, Geometry and Donuts



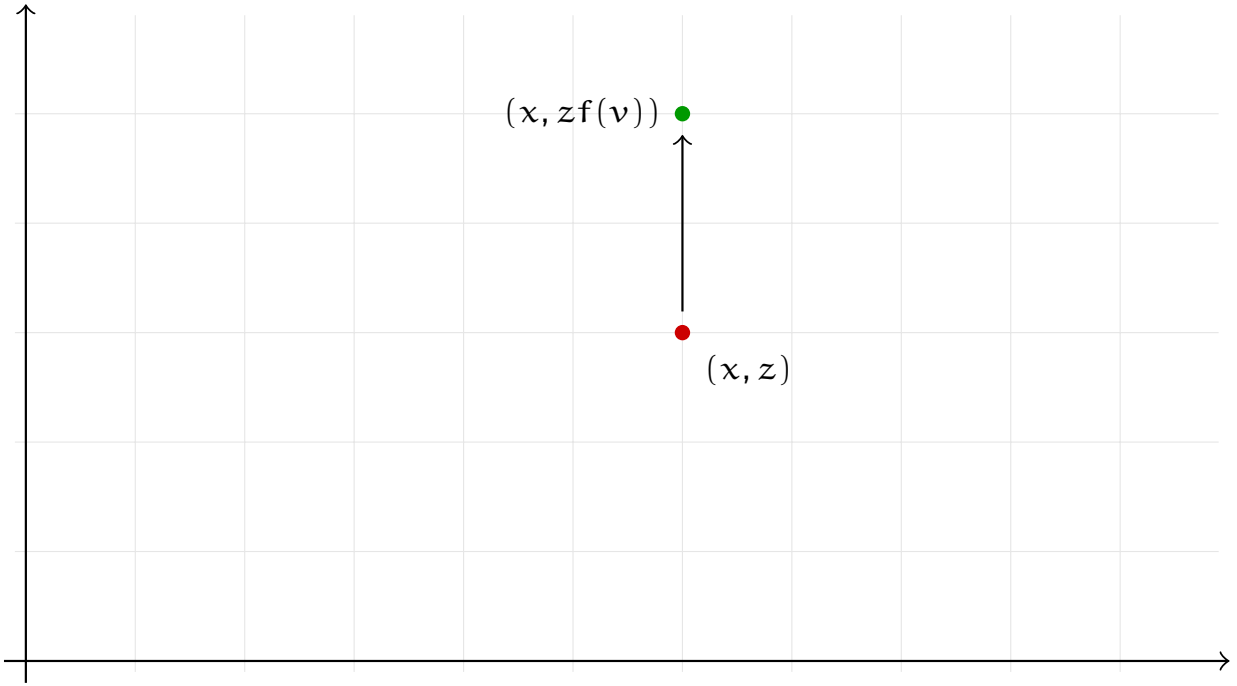
The generator g and the encryption key y_1 describe a point and a line through the origin.

ElGamal, Geometry and Donuts



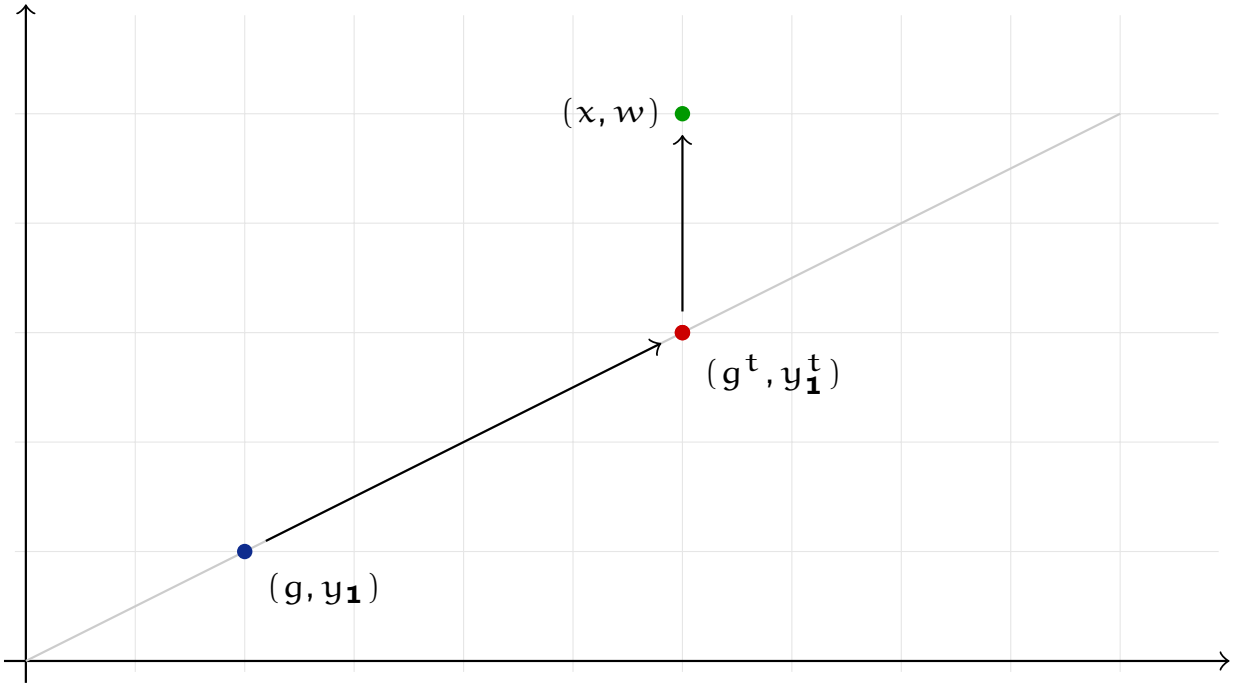
For any integer t, the points (g, y_1) and (g^t, y_1^t) lie on the same line through the origin.

ElGamal, Geometry and Donuts



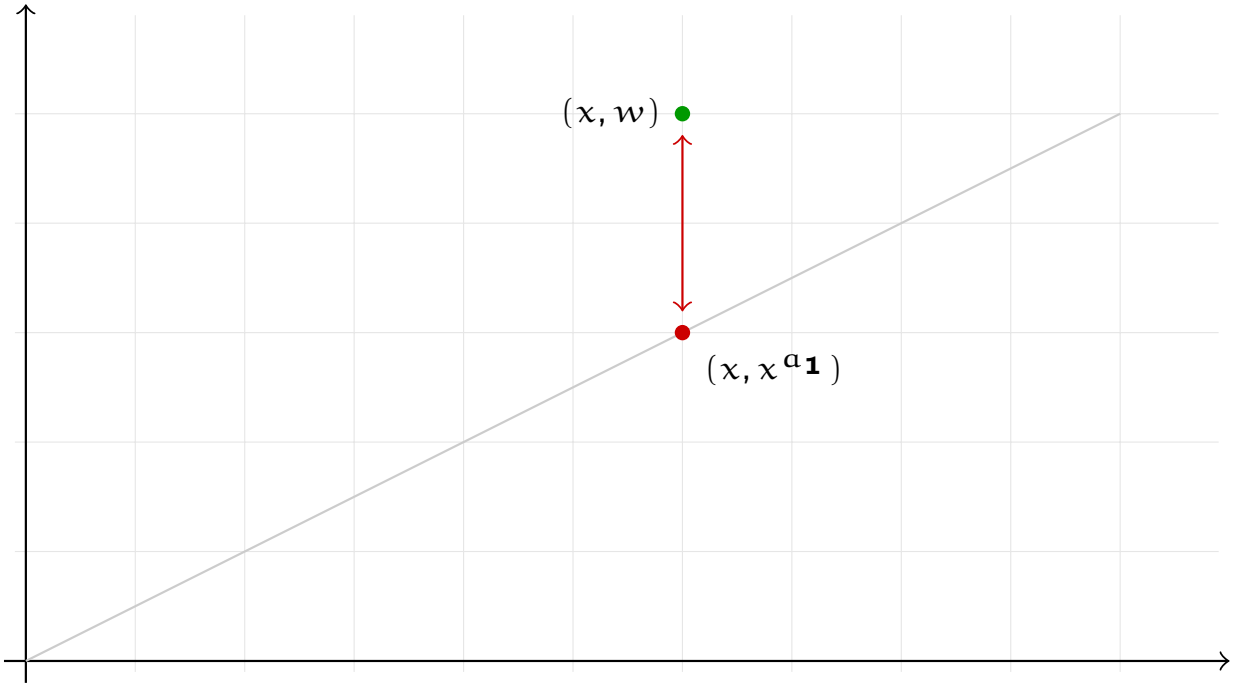
Multiplying $f(v)$ into the second coordinate amounts to a vertical shift.

ElGamal, Geometry and Donuts



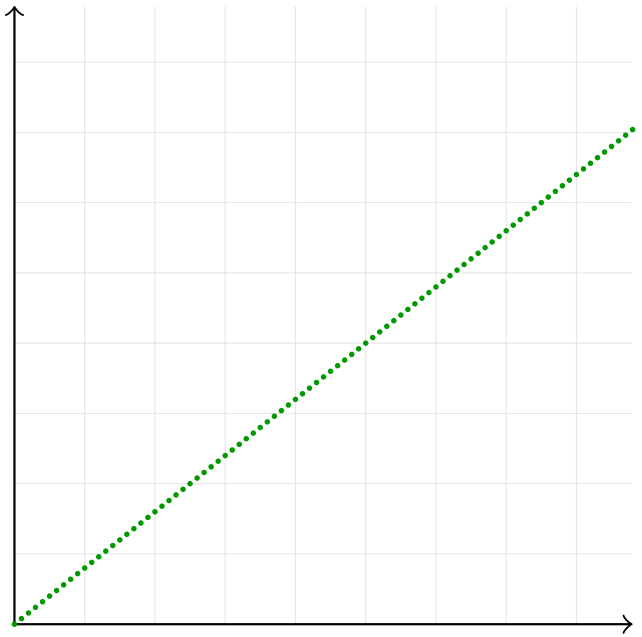
ElGamal encryption: (i) choose a random point on the line defined by the encryption key, and (ii) shift the point vertically by the message.

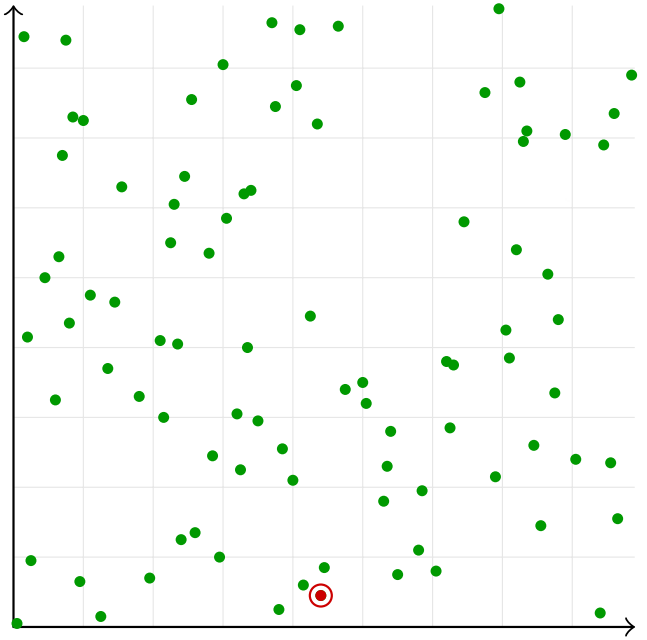
ElGamal, Geometry and Donuts



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.

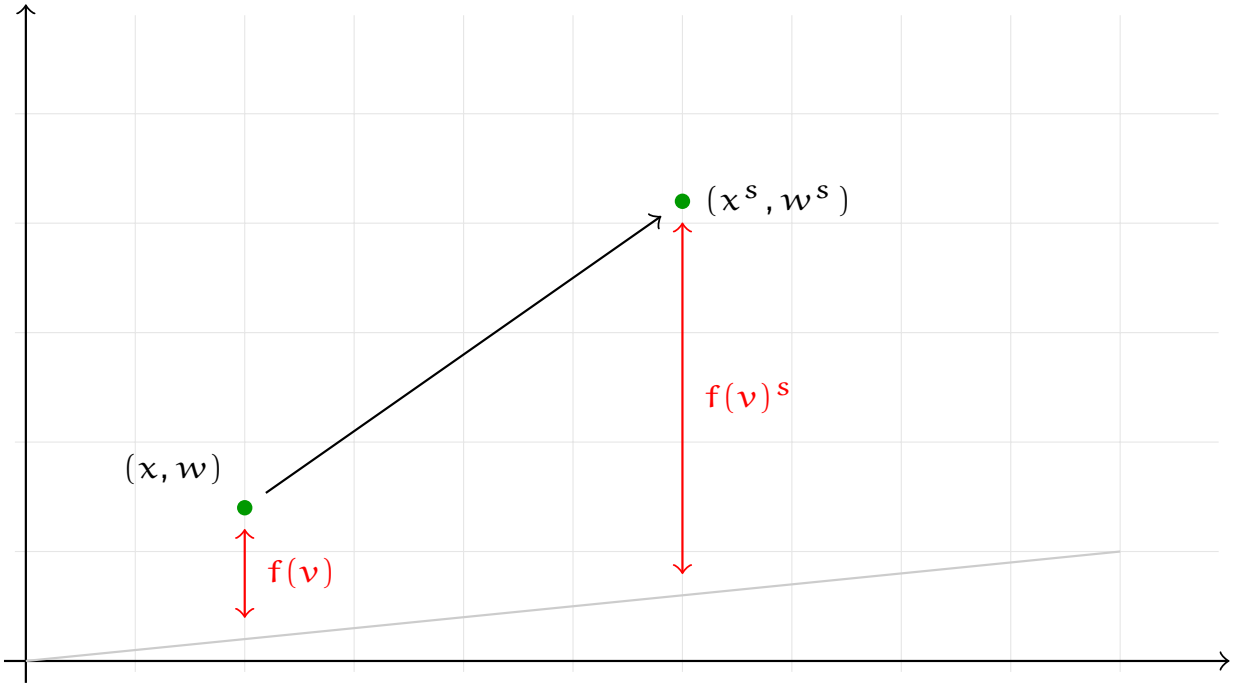
ElGamal, Geometry and Donuts





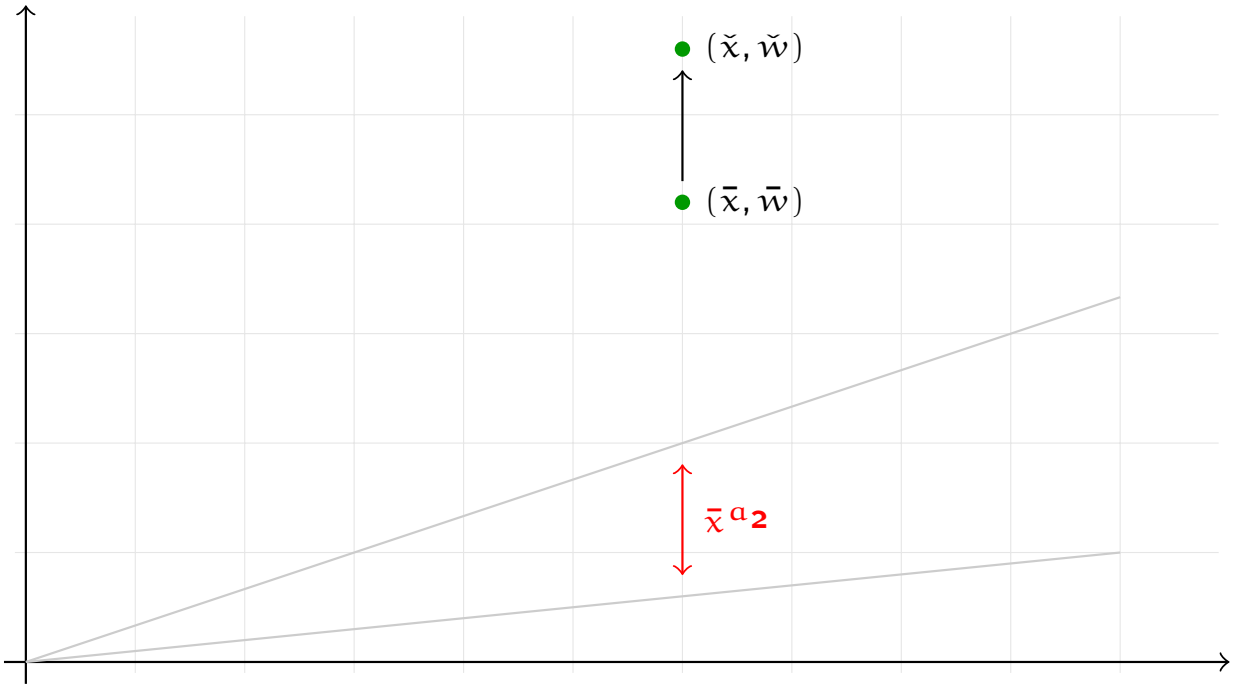
Decision Diffie-Hellman Assumption: Given two points, it is hard to say anything about the vertical distance between the second point and the line defined by the first point, unless you know the slope of the line.

ElGamal, Geometry and Donuts



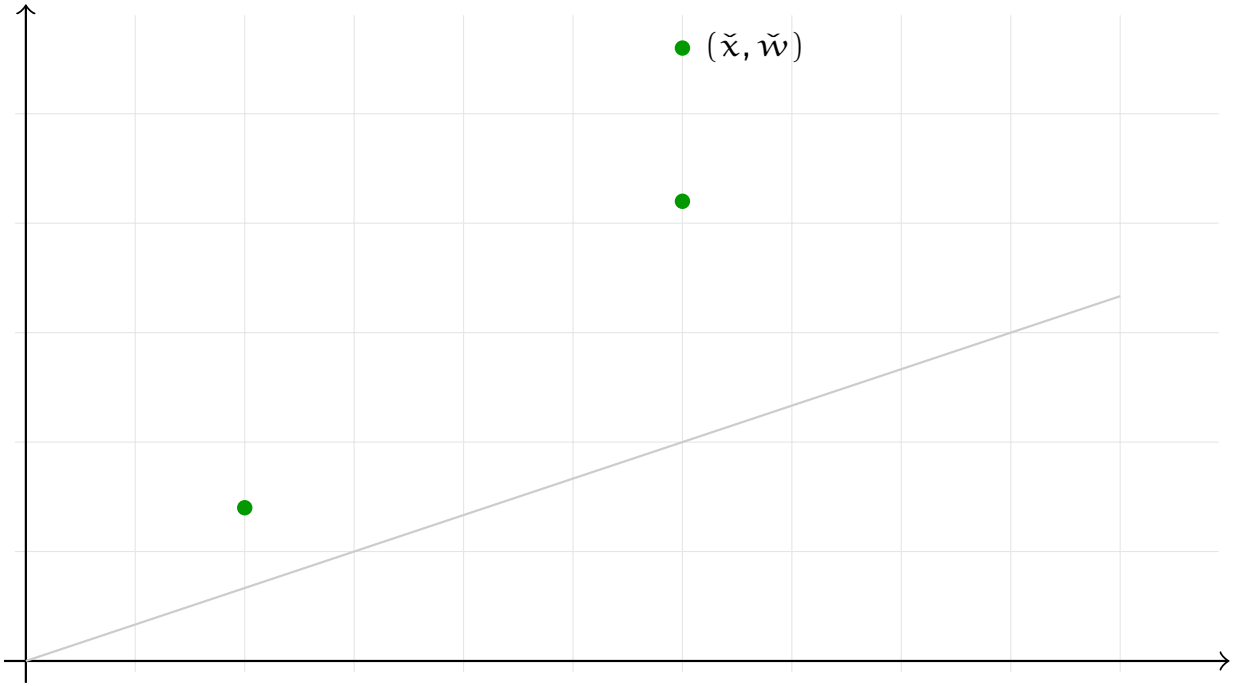
The ballot box knows the per-voter exponent s and computes a new ciphertext $(\bar{x}, \bar{w}) = (x^s, w^s)$.

ElGamal, Geometry and Donuts



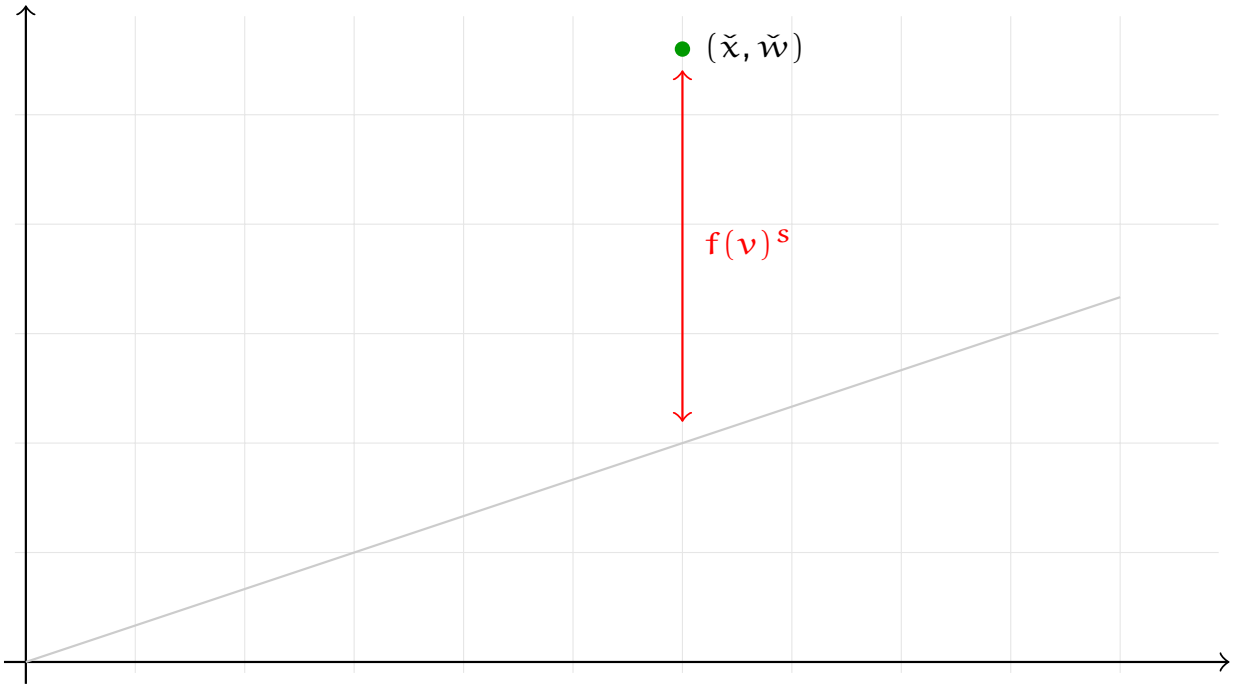
The ballot box knows the *difference* a_2 between two slopes. It uses this to compute a new ciphertext $(\check{x}, \check{w}) = (\bar{x}, \bar{w}\bar{x}^{a_2})$.

ElGamal, Geometry and Donuts



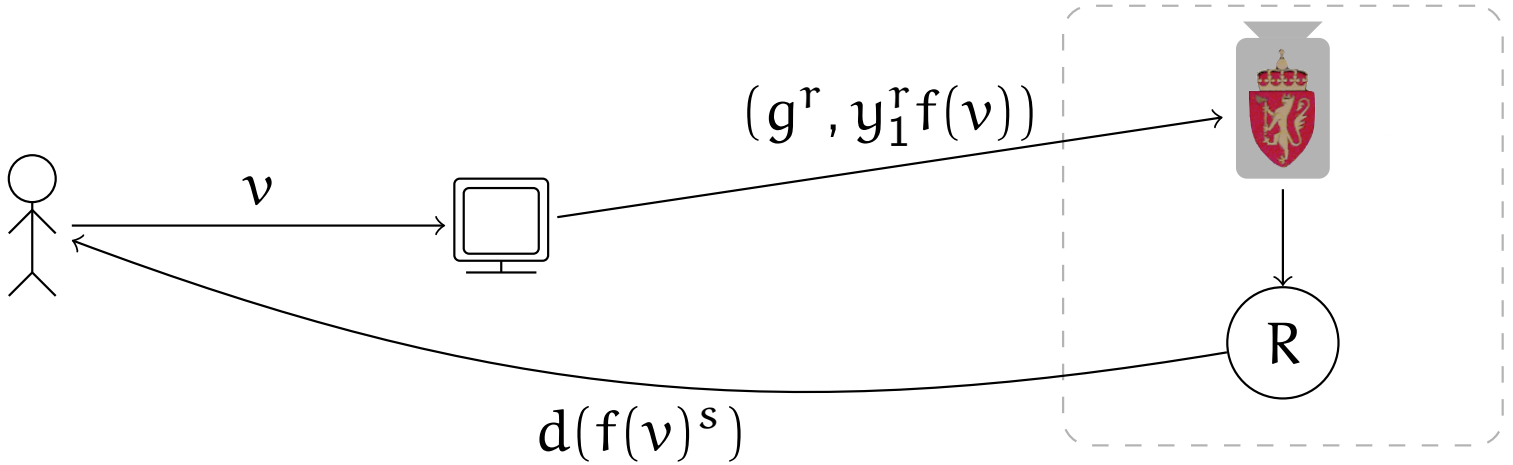
The receipt generator gets all three ciphertexts.

ElGamal, Geometry and Donuts



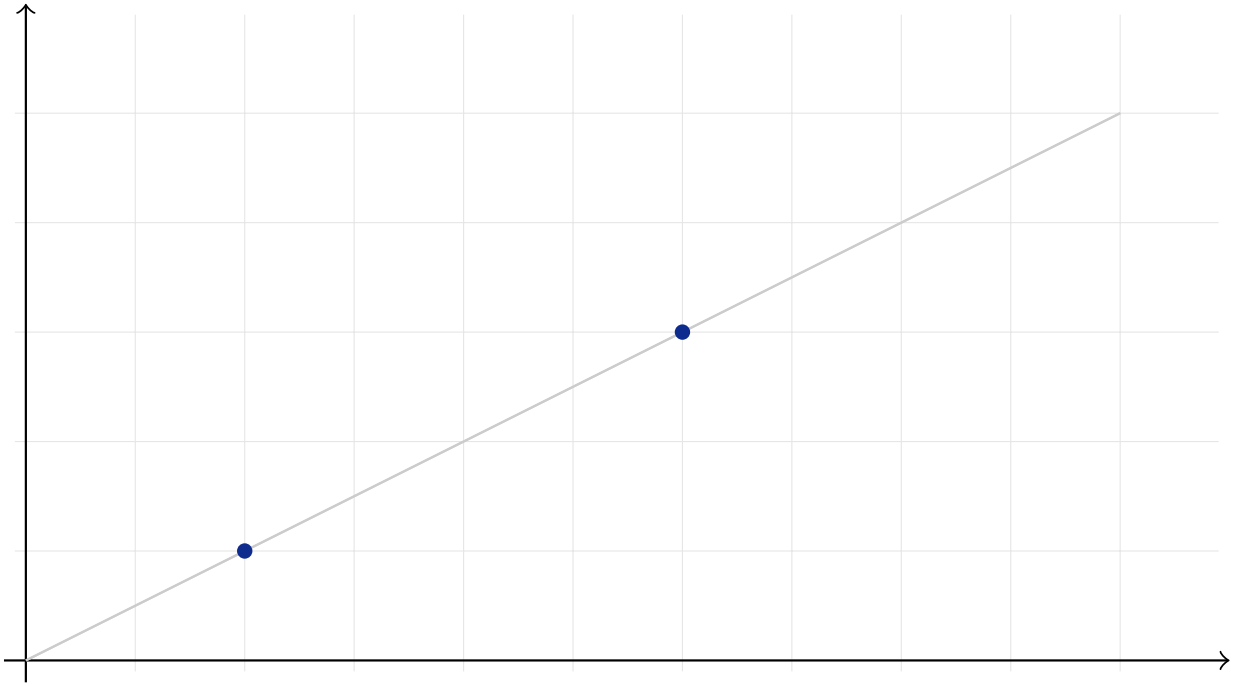
The receipt generator knows the slope of the second line and can decrypt only the final ciphertext.

Computing the Receipt Code



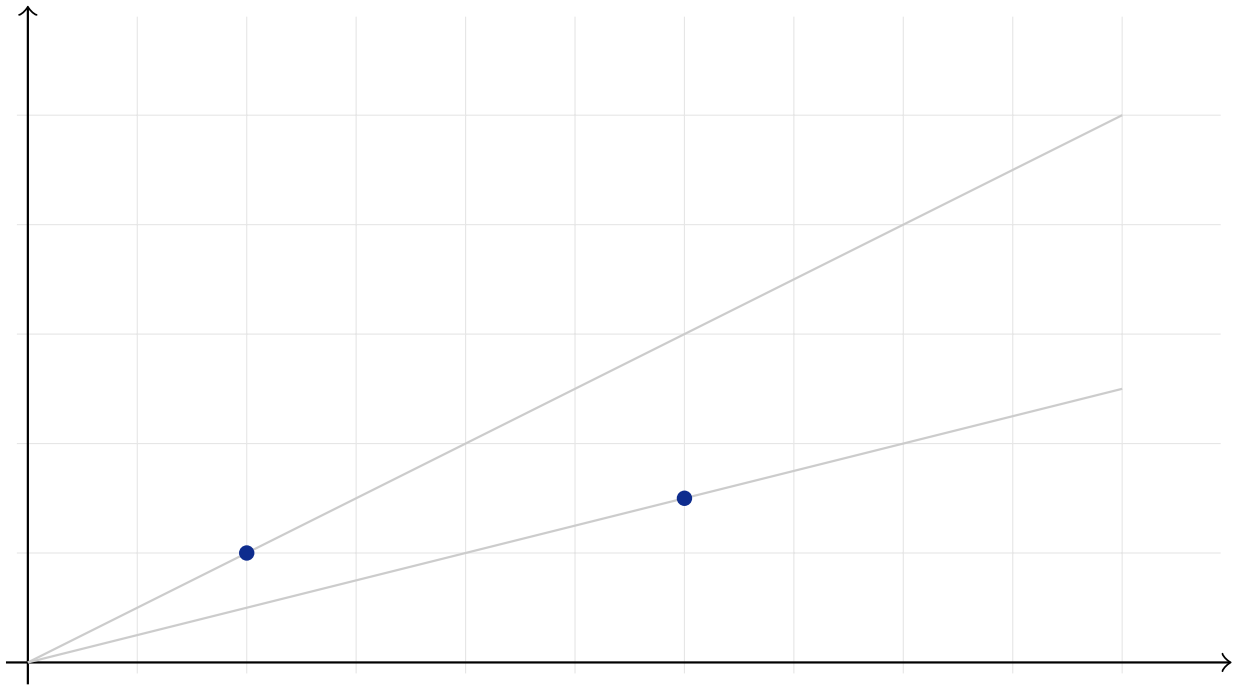
Security: (i) The ballot box sees nothing because of ElGamal, and (ii) the receipt generator sees nothing because $f(v)^s$ looks random. Does it?

Encoding Options in the Group



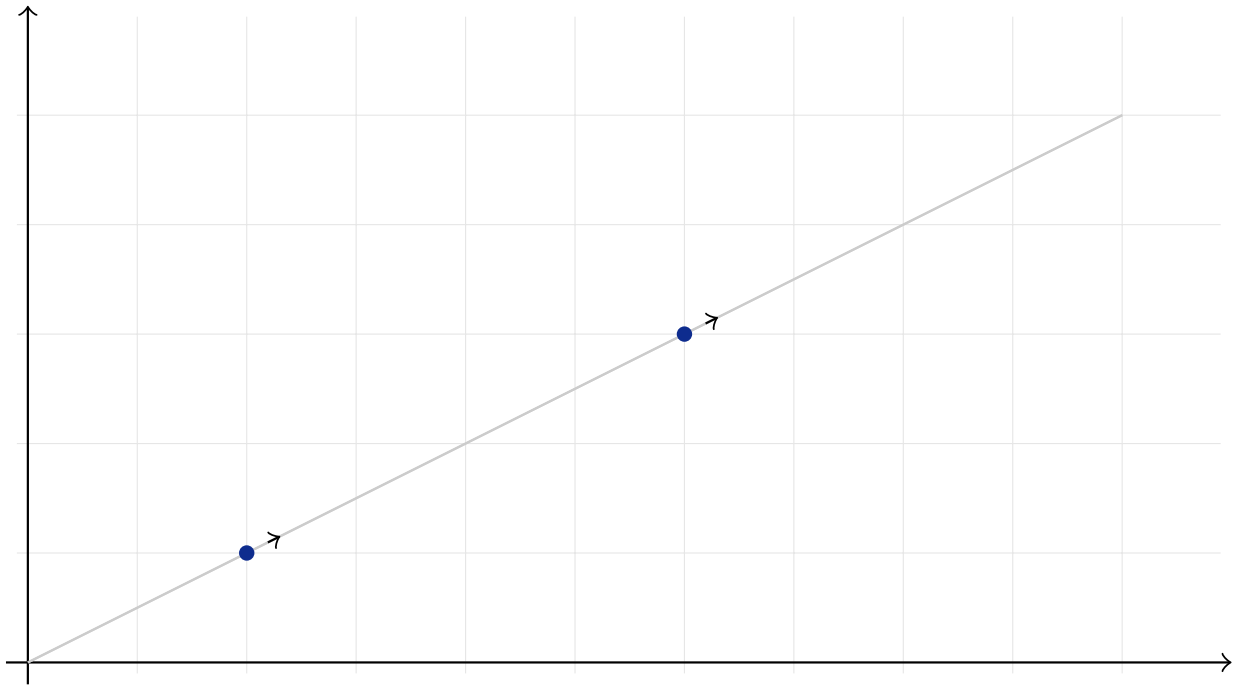
Given two points in the plane, DDH says it is hard to tell if they lie on the same line through the origin.

Encoding Options in the Group



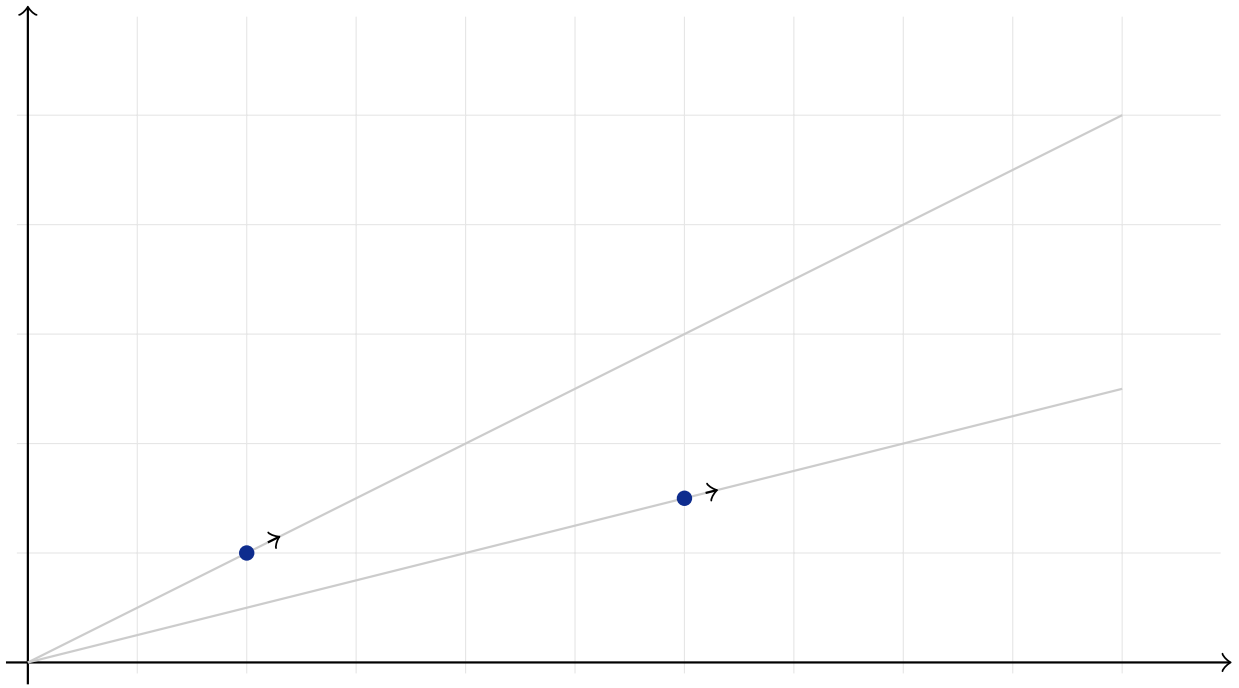
Given two points in the plane, DDH says it is hard to tell if they lie on the same line through the origin, or on different lines.

Encoding Options in the Group



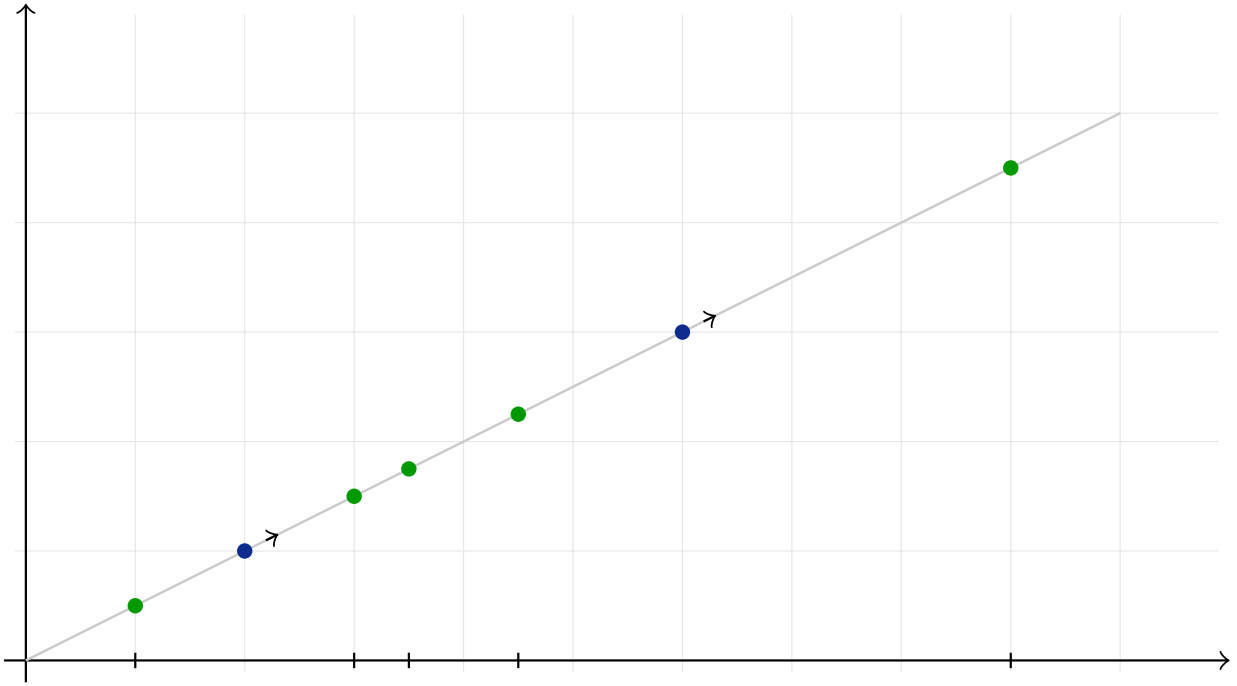
We can make random linear combinations of two points. If the points lie on the same line, the linear combination stays on the line.

Encoding Options in the Group



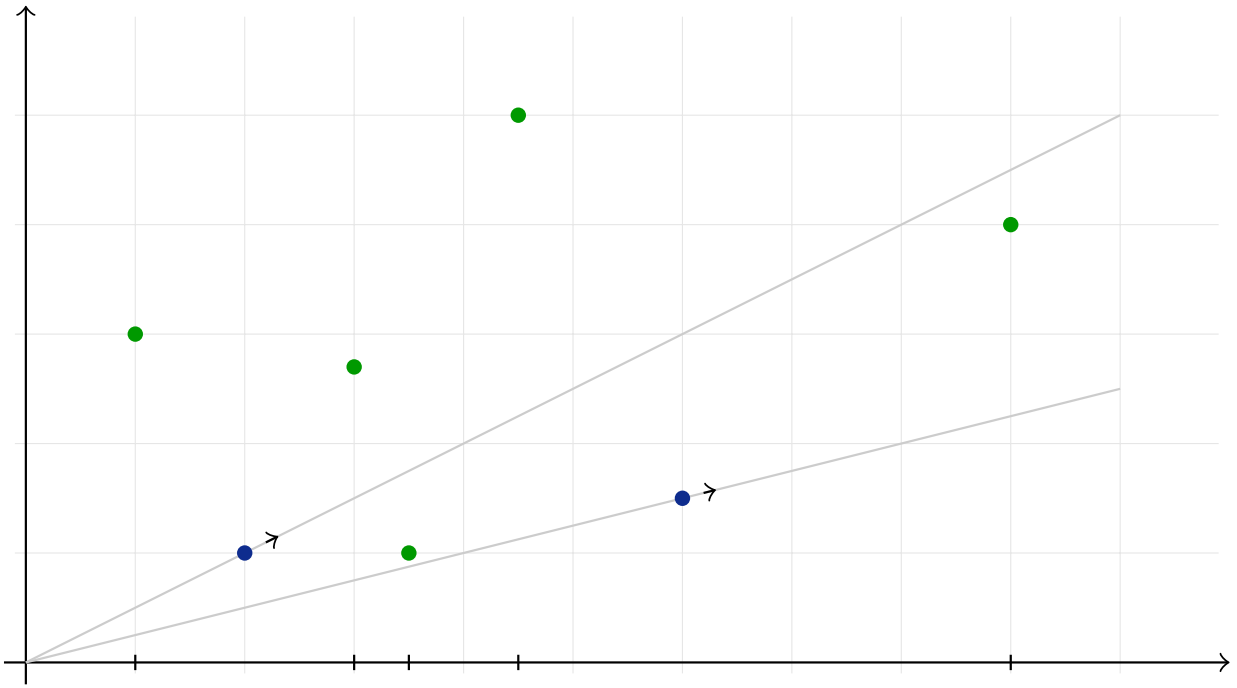
We can make random linear combinations of two points. If the points lie on different lines, the linear combination could end up anywhere.

Encoding Options in the Group



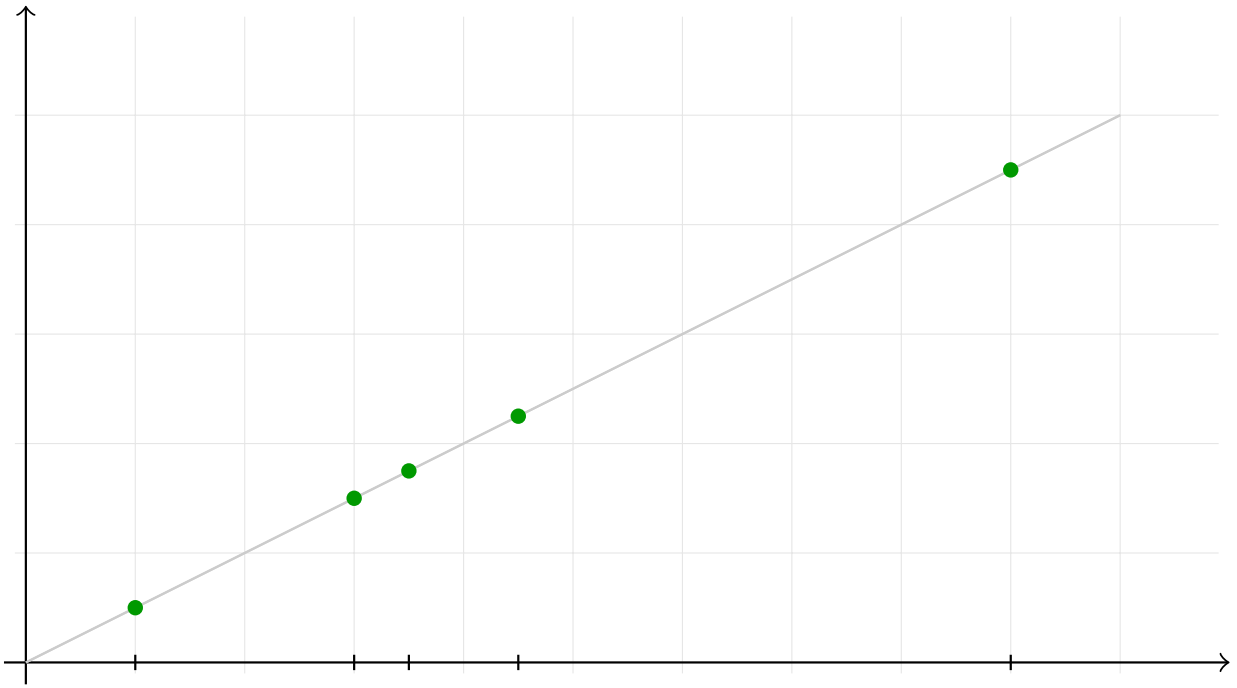
Making many linear combinations, we get either a many colinear points.

Encoding Options in the Group



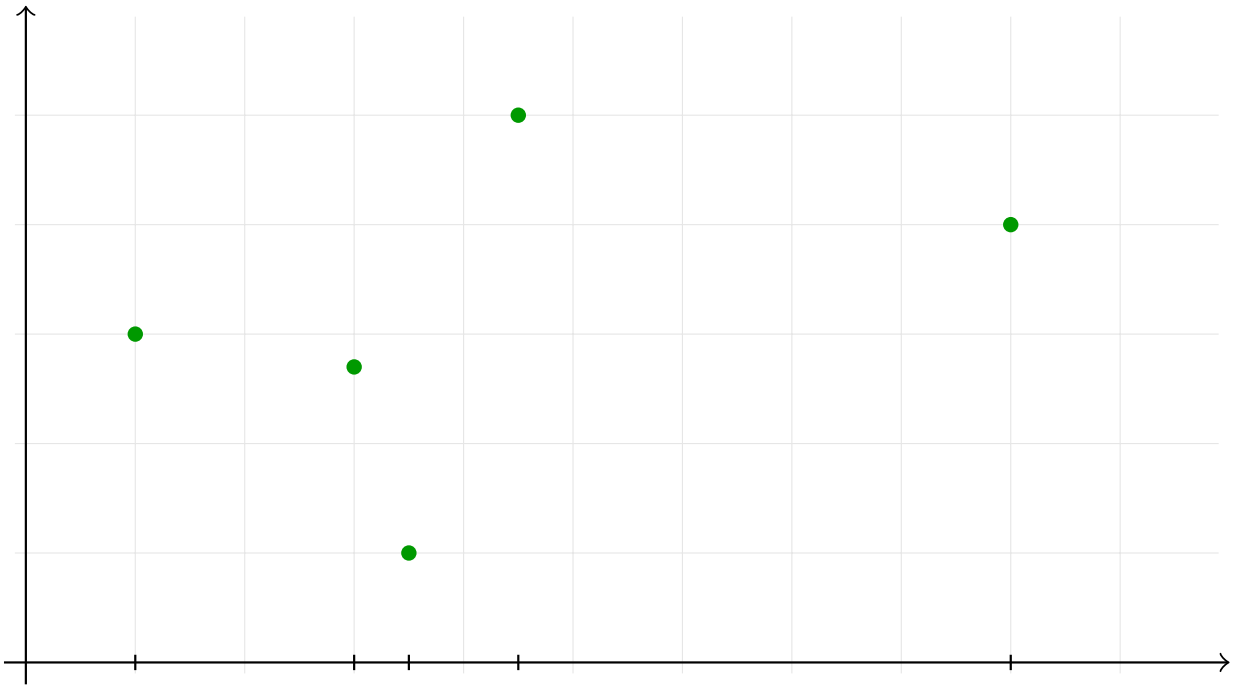
Making many linear combinations, we get either a many colinear points, or many random points.

Encoding Options in the Group



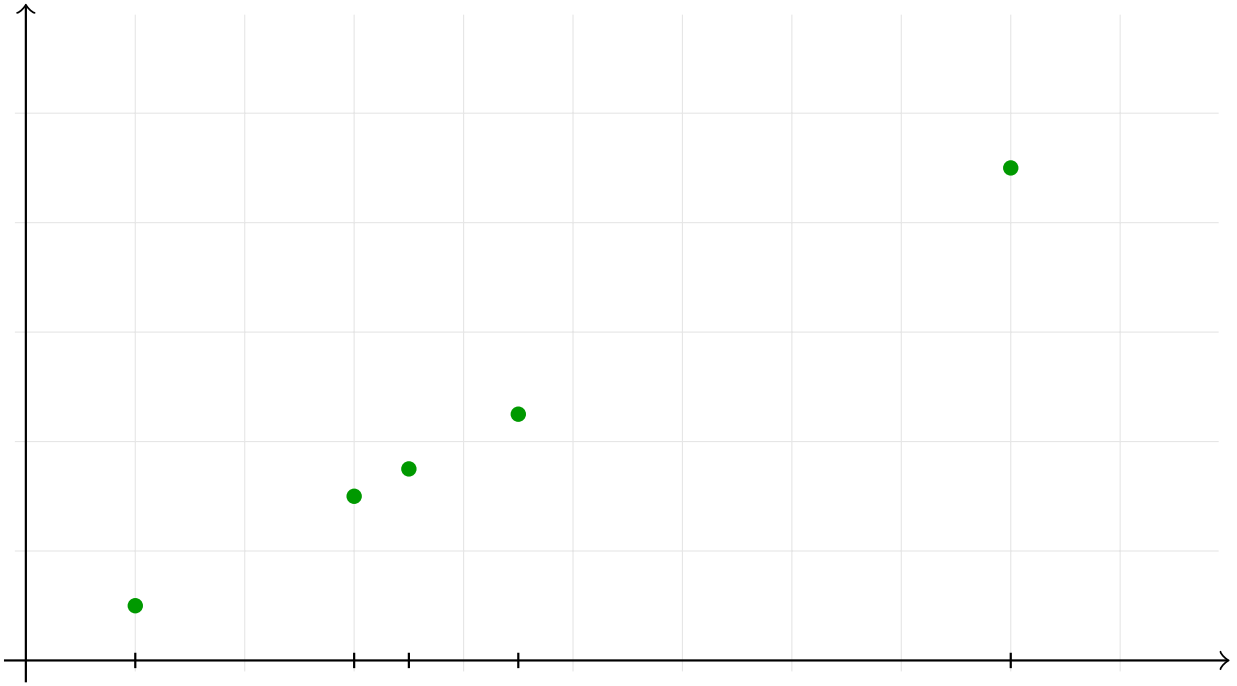
In other words, if $f : \{\text{option set}\} \to G$ is a random function, we cannot distinguish between $v \mapsto f(v)^s$ and a random function.

Encoding Options in the Group



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 Options in the Group



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

Technique: Ideal functionalities

Complex cryptographic protocols typically use several subprotocols, who may also be very complex. To simplify analysis, we would like to abstract away the subprotocols.

We use a variant of Canetti's universal composability framework, where so-called ideal functionalities are used as abstractions for subprotocols.

> *Any security we can establish for the protocol using ideal functionalities, should still hold for the full protocol.*

Ideal Electronic Identity

The voter informs the ideal identity functionality that he wants to use a computer. Afterwards, the computer may use the ideal functionality to connect to servers or sign documents on behalf of the voter.

Note (Berry)

This provides secure authenticated channels and an external authentication primitive.

Ideal Electronic Identity

The voter informs the ideal identity functionality that he wants to use a computer. Afterwards, the computer may use the ideal functionality to connect to servers or sign documents on behalf of the voter.

— Note that the voter uses his electronic identity all the time, not just for voting.

Ideal Electronic Identity

The voter informs the ideal identity functionality that he wants to use a computer. Afterwards, the computer may use the ideal functionality to connect to servers or sign documents on behalf of the voter.

— A compromised computer will be able to do whatever on behalf of the voter. This usually gives the attacker slightly more power than in the real world.

Ideal Electronic Identity

The voter informs the ideal identity functionality that he wants to use a computer. Afterwards, the computer may use the ideal functionality to connect to servers or sign documents on behalf of the voter.

— Typically realized by normal digital signatures, TLS and some PKI. Other more creative solutions exist.

Proofs of Knowledge

1. The players provide pairs of group elements and corresponding discrete logarithms to the ideal functionality. The functionality replies with a «proof» and records the proof and the group elements.
2. When some other player wants to verify a «proof», the ideal functionality checks its records.

— Realized by non-interactive Schnorr proofs.

Proofs of Knowledge

1. The players provide pairs of group elements and corresponding discrete logarithms to the ideal functionality. The functionality replies with a «proof» and records the proof and the group elements.
2. When some other player wants to verify a «proof», the ideal functionality checks its records.

— Realized by non-interactive Schnorr proofs.

— By proving existence of discrete logarithms in $\mathrm{G} \times \mathrm{G}$ or $\mathrm{G} \times \mathrm{G} \times \mathrm{G}$, we can prove equality of discrete logarithms in G.

Proofs of Knowledge

1. The players provide pairs of group elements and corresponding discrete logarithms to the ideal functionality. The functionality replies with a «proof» and records the proof and the group elements.
2. When some other player wants to verify a «proof», the ideal functionality checks its records.

— Realized by non-interactive Schnorr proofs.

Note (Berry)

This is zero knowledge.

Proofs of Knowledge

1. The players provide pairs of group elements and corresponding discrete logarithms to the ideal functionality. The functionality replies with a «proof» and records the proof and the group elements.
2. When some other player wants to verify a «proof», the ideal functionality checks its records.

— Realized by non-interactive Schnorr proofs.

Note

Schnorr is not extractable when composed in parallel. This seems to be a technical obstruction. Therefore, we pretend Schnorr is extractable to make the proof go through.

Key Generation

The ideal functionality provides players with randomly chosen keys.

- Keys are generated by the electoral board using standard multiparty computation.
- Here, we abstract away creating and distributing (by mail!) voting cards to voters and distributing election keys to computers.
- A sufficiently large honest majority allows us to assume honestly sampled keys that would be known to a simulator.
- Could be made information-theoretically secure. . .

Out-of-band Communication

The ideal functionality allows the receipt generator to send messages directly to the voter.

Out-of-band Communication

The ideal functionality allows the receipt generator to send messages directly to the voter.

Note (Berry)

Remember that untappable channels are a warning sign.

Out-of-band Communication

The ideal functionality allows the receipt generator to send messages directly to the voter.

- Realized by SMS messages.
- Smartphones are generally connected to computers. If the computer is compromised, the phone may also be compromised. Example: Jailbreaking the iPhone.

Out-of-band Communication

The ideal functionality allows the receipt generator to send messages directly to the voter.

- Realized by SMS messages.
- Smartphones are generally connected to computers. If the computer is compromised, the phone may also be compromised. Example: Jailbreaking the iPhone.
- As a side effect, the voter is notified when the infrastructure receives a ballot. This means the computer cannot undetectably submit ballots on the voter's behalf.

The Protocol: Voter

Before voting, the voter must have received his voting card.

1. The voter gives the computer custody of his electronic identity.
2. He tells the computer what ballot to submit.
3. He waits for a receipt from the computer, and the receipt codes via the mobile phone.
4. He verifies only the receipt codes.
5. If everything works out, he accepts the ballot as cast and erases the SMS message. Otherwise he votes on paper.

If the voter unexpectedly receives a receipt code SMS, he complains (and votes on paper).

The Protocol: Computer

1. The computer receives custody of the voters electronic identity.
2. It receives a ballot to submit.
3. It encrypts the ballot as discussed, generates a proof of knowledge for the ciphertexts and signs the encrypted ballot on behalf of the voter. The result is sent to the ballot box.
4. When the ballot box replies with a receipt, the computer verifies that the receipt is a signature on a hash of the encrypted ballot.
5. The computer presents the receipt to the voter.

The Protocol: Ballot Box

1. The ballot box receives an encrypted ballot, a proof of knowledge and a signature.
2. The signature and the proof of knowledge are verified.
3. The next sequence number is chosen. The receipt code ciphertexts are generated with correctness proofs. Everything is sent to the receipt generator.
4. The ballot box waits for a receipt from the receipt generator. It verifies that the receipt is a signature on a hash of the encrypted ballot.
5. The sequence number, voter identity, encrypted ballot, proof of knowledge and digital signature is recorded. The receipt is forwarded to the computer.

The Protocol: Ballot Box

When told to start counting:

1. Exclude voters that voted on paper.
2. For every remaining voter, select the record with maximal sequence number.
3. Multiply all the ciphertexts from the encrypted ballot into a single ciphertext.
4. List the resulting ciphertexts in «random canonical» order and send the result for decryption.
5. Send the entire ballot box content to the auditor.
6. Stop.

The Protocol: Receipt Generator

1. The receipt generator receives a sequence number, encrypted ballot, proof of knowledge, receipt code ciphertexts and proofs of correctness from the ballot box.
2. It verifies that the sequence number is maximal for this voter and that the encrypted ballot and proof of knowledge is fresh.
3. It verifies the signature, the proof of knowledge and the proofs of correctness.
4. The receipt codes are computed and sent to the voter.
5. A hash of the encrypted ballot, proof of knowledge and signature is signed and returned to the ballot box.

The Protocol: Receipt Generator

When told to start counting:

1. Send the hashes of everything seen to the auditor.
2. Stop.

The Protocol: Decryption Service

1. Wait for a list of ciphertexts from the ballot box.
2. Send a hash of the list to the auditor.
3. Shuffle the ciphertexts and decrypt them.
4. Send the decrypted ballots for counting.
5. Send the decrypted ballots to the auditor and prove to him that the input ciphertexts contain a permutation of the decrypted ballots.
6. Stop.

The Protocol: Decryption Service

1. Wait for a list of ciphertexts from the ballot box.
2. Send a hash of the list to the auditor.
3. Shuffle the ciphertexts and decrypt them.
4. Send the decrypted ballots for counting.
5. Send the decrypted ballots to the auditor and prove to him that the input ciphertexts contain a permutation of the decrypted ballots.
6. Stop.

Note

I have not (yet) analysed the correctness proof for the shuffle.

The Protocol: Auditor

1. The auditor receives the ballot box contents, hashes from the receipt generator and a hash, a list of ballots and a shuffle proof from the decryption service.
2. Verify that the receipt generator saw the ballot box content.
3. Verify that the decryption service got the correct ciphertexts.
4. Verify the decryption service shuffle proof.
5. Send the ballot list to the election board. Publish the hashes received from the receipt generator.
6. Stop.

The Protocol: Auditor

1. The auditor receives the ballot box contents, hashes from the receipt generator and a hash, a list of ballots and a shuffle proof from the decryption service.
2. Verify that the receipt generator saw the ballot box content.
3. Verify that the decryption service got the correct ciphertexts.
4. Verify the decryption service shuffle proof.
5. Send the ballot list to the election board. Publish the hashes received from the receipt generator.
6. Stop.

Note

I have not (yet) analysed the correctness proof for the shuffle.

The Protocol: Auditor

1. The auditor receives the ballot box contents, hashes from the receipt generator and a hash, a list of ballots and a shuffle proof from the decryption service.
2. Verify that the receipt generator saw the ballot box content.
3. Verify that the decryption service got the correct ciphertexts.
4. Verify the decryption service shuffle proof.
5. Send the ballot list to the election board. Publish the hashes received from the receipt generator.
6. Stop.

Note II

We can have multiple auditors.

The Protocol: Auditor

1. The auditor receives the ballot box contents, hashes from the receipt generator and a hash, a list of ballots and a shuffle proof from the decryption service.
2. Verify that the receipt generator saw the ballot box content.
3. Verify that the decryption service got the correct ciphertexts.
4. Verify the decryption service shuffle proof.
5. Send the ballot list to the election board. Publish the hashes received from the receipt generator.
6. Stop.

Note II

We can have multiple auditors. But we cannot use political parties, because anyone who sees the final ballots can reliably buy votes.

Threat Model & Security Goal

Security goal:

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

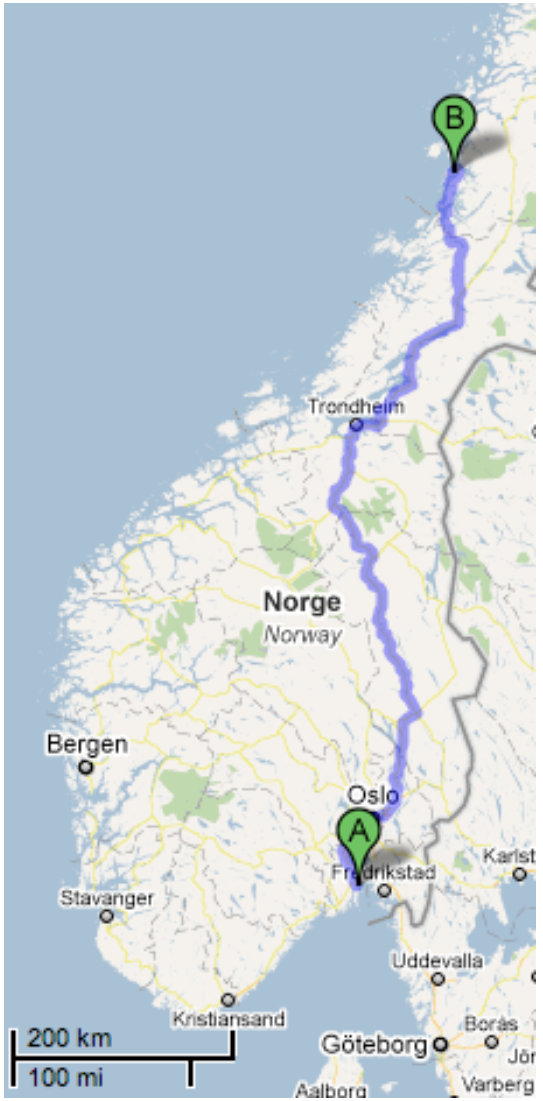
Threat model:

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

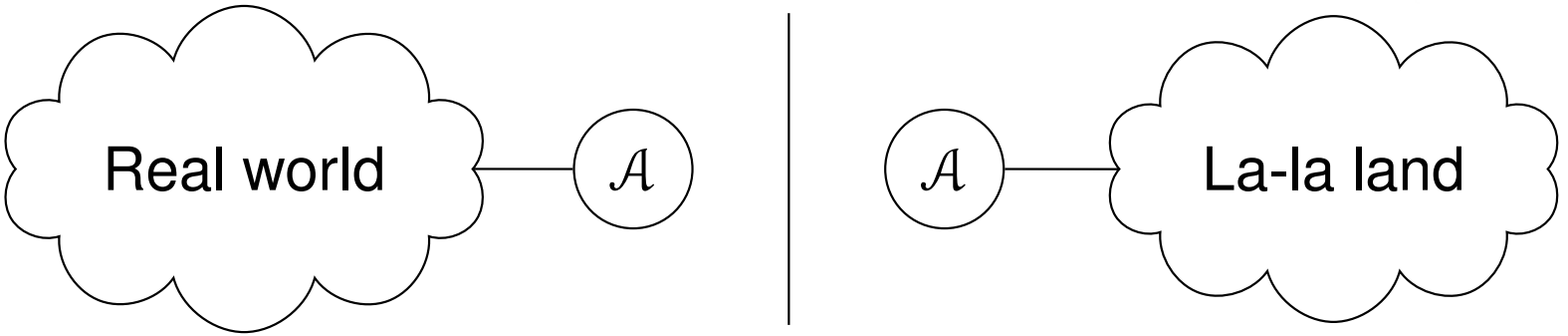
A The Receipt Generator

B The Ballot Box



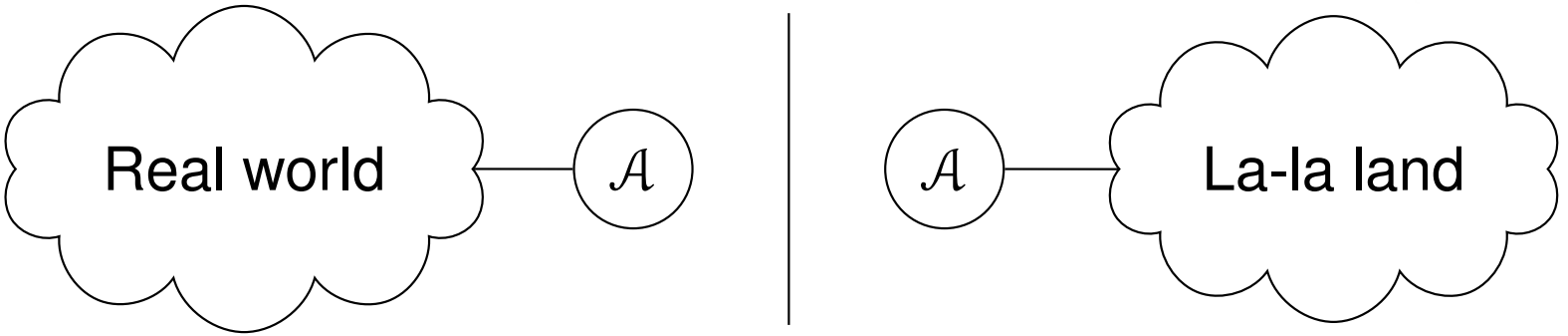
Technique: Games People Play

An attacker is an algorithm that interacts with its surroundings in a well-defined manner.



Technique: Games People Play

An attacker is an algorithm that interacts with its surroundings in a well-defined manner.

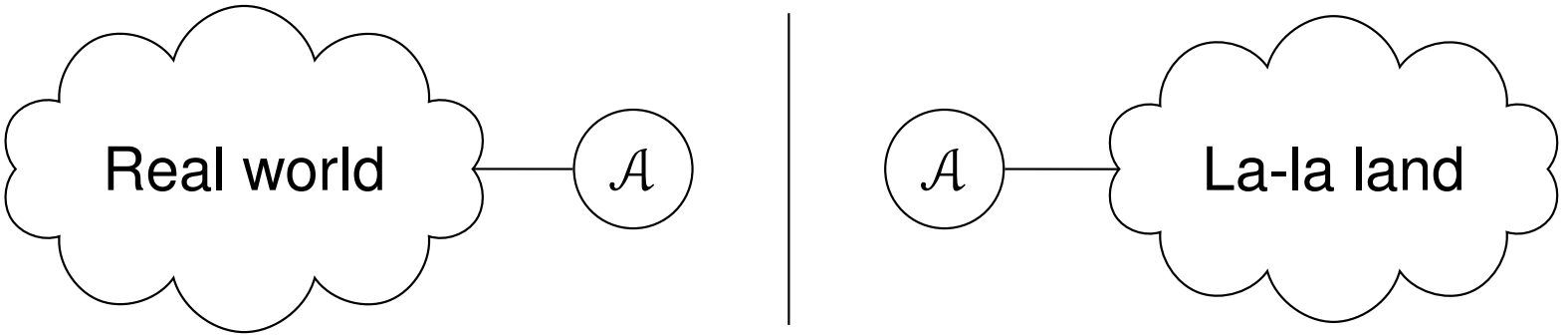


Idea

If no attacker can distinguish the real world from la-la land, and we can prove that something is true in la-la land, that must also be true in the real world.

Technique: Games People Play

An attacker is an algorithm that interacts with its surroundings in a well-defined manner.

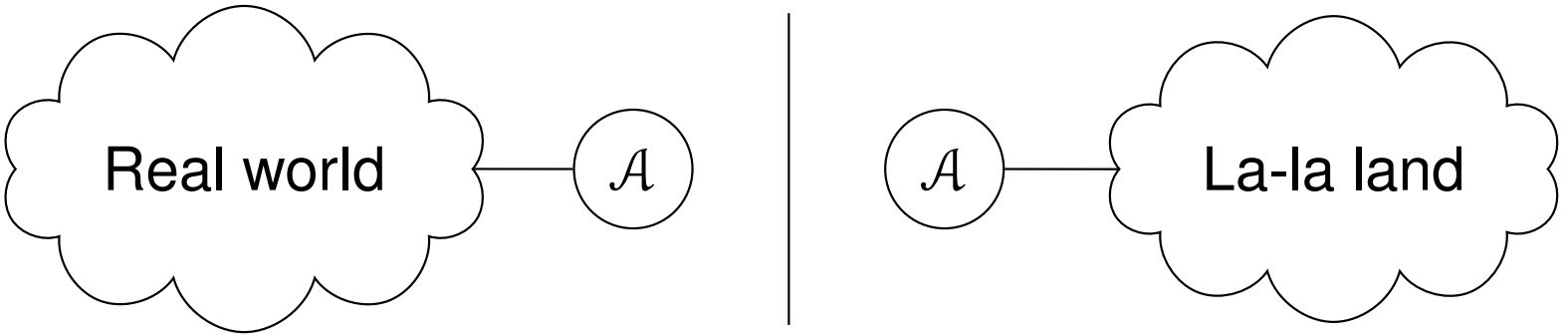


Idea II

We can design many la-la lands.

Technique: Games People Play

An attacker is an algorithm that interacts with its surroundings in a well-defined manner.

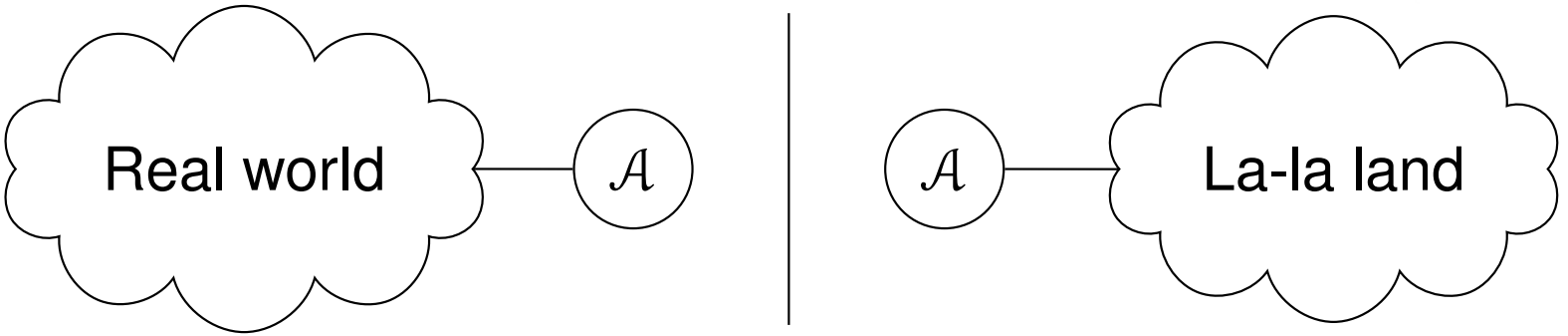


Idea III

By making very small, cumulative changes to the real world, we get a sequence of la-la lands. If we prove that each consecutive la-la land is indistinguishable from the previous one, it follows that the real world must be indistinguishable from the final la-la land.

Technique: Games People Play

An attacker is an algorithm that interacts with its surroundings in a well-defined manner.

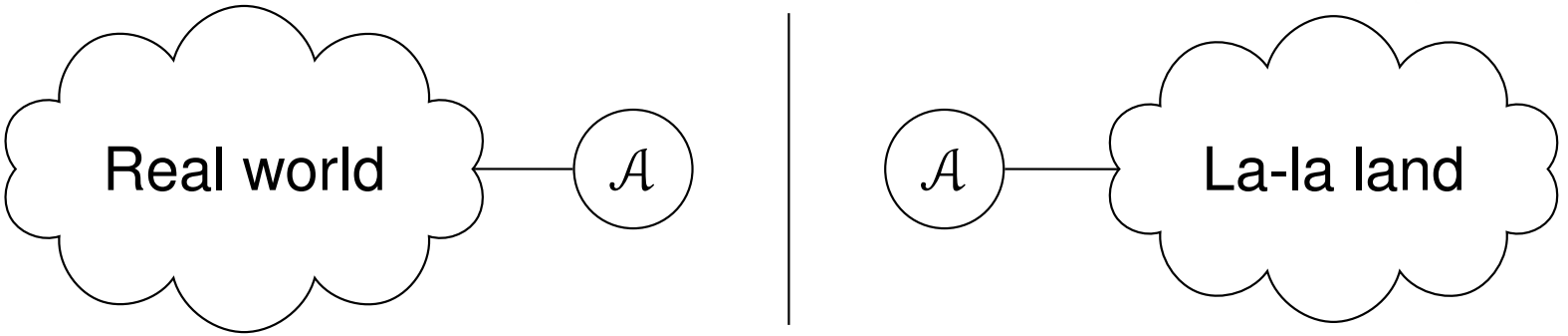


Idea IV

We want to end up in a la-la land where security holds trivially.

Technique: Games People Play

An attacker is an algorithm that interacts with its surroundings in a well-defined manner.



By the way

We do not say la-la land, we say *games*.

Corrupt: Ballot Box/Voters/Computers

Game 0

Real World.

Corrupt: Ballot Box/Voters/Computers

Game 1:

Changes: All honest players and ideal functionalities are played by one simulator.

Argument: Obvious.

Observation: Our simulator knows every secret key.

Corrupt: Ballot Box/Voters/Computers

Game 2:

Changes: Abort if a hash collision is observed.

Argument: Our hash function is collision resistant, therefore we will never abort, therefore it is impossible to distinguish the games.

Observation: We can now treat the hash function as injective.

Corrupt: Ballot Box/Voters/Computers

Game 3:

Changes: Let the per-voter pseudo-random function be a real random function.

Argument: It's a pseudo-random function, it's what they do.

Corrupt: Ballot Box/Voters/Computers

Game 4:

Changes: Compute receipt codes as $\mathrm{d}(\mathrm{f}(v))$ instead of $\mathrm{d}(\mathrm{f}(v)^s)$. The simulated receipt generator decrypts the encrypted ballot instead of the receipt code ciphertexts.

Argument: A random function composed with a permutation is a random function, so nothing is observable unless the decryption changes. But that can't happen because we have sound proofs of correctness.

Observation: The receipt code decryption key is no longer used.

Corrupt: Ballot Box/Voters/Computers

Game 5:

Changes: The receipt generator and decryption service no longer decrypt honestly generated encrypted ballots, but instead use the ballot directly.

Argument: Obvious.

Corrupt: Ballot Box/Voters/Computers

Game 6:

Changes: We pretend that the Schnorr proof is extractable. Adversarially generated encrypted ballots are decrypted using a witness extracted from the Schnorr proof, not the election decryption key.

Argument: Every ciphertext that arrives for decryption must have been seen by the recept generator, where we saw the Schnorr proof and extracted the witness.

Observation: The election decryption key is no longer used.

Corrupt: Ballot Box/Voters/Computers

Game 7:

Changes: We fake the honestly generated proofs of knowledge.

Argument: Technical.

Corrupt: Ballot Box/Voters/Computers

Game 8:

Changes: The simulator stops encrypting real votes, instead encrypts nonsense.

Argument: Decision Diffie-Hellman.

Corrupt: Ballot Box/Voters/Computers

Observations:

- The encrypted ballots never contain any information about the real ballots unless the computer is corrupt. The final output ballots have been shuffled.
- The auditor ensures that every counted ballot has been seen by the receipt generator.
- The voter will not accept the ballot as cast unless the receipt generator sees a ballot that with high probability equals the intended ballot.
- Sequence numbers ensure that the final ballot counts.

Corrupt: Ballot Box/Voters/Computers

Note

The protocol is not perfect. There are somewhat subtle attacks that allow an active attacker to guess votes.

Honest: PC/Auditor

What if voter's computer and the auditor are the only honest players?

Conjecturally: The auditor publishes hashes of every record in the ballot box (without sequence numbers). The voter's computer downloads this list, verifies that its hash is on the list. If it isn't, the signature allows the voter to complain convincingly.

NB!

This has not been analysed yet.

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 . . .