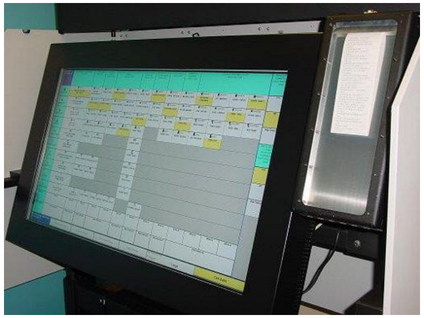
Advanced Web Security

Electronic Voting

Electronic Voting

- Refers to two things
  - Electronic device is used to collect votes
  - Voting over Internet using e.g., computer or smart phone
- DRE machines (Direct Recording Electronic)



- We focus on Internet voting

Voting Phases

- **Voter Registration Phase** – All eligible voters are registered as voters.
- **Voting Phase** – All registered voters are allowed to cast their vote.
- **Tallying Phase** – The votes are counted to obtain the final result

Voting Properties

- **Privacy/Anonymity** – It should be impossible for anyone to extract any information about someone else's vote.
- **Correctness** – The result of the election matches the intention of the voters
- **Verifiability**
  - **Individual Verifiability** – It should be possible for voters to ensure that their vote was recorded as intended and included in the computation of the final result.
  - **Universal Verifiability** – It should be possible for a third party to ensure that all votes have been included in the computation of the final result and that the election was properly performed
- **Voter Eligibility** – Only voters that are allowed to vote can vote
- **One-Voter-One-Vote** – It should not be possible to vote twice

More Voting Properties

- **Receipt-Freeness** – It should not be possible for a voter to prove how he/she votes
- **Coercion-Resistance** – It should not be possible to coerce someone to vote in a particular way
- **Robustness/Fault Tolerance** – Some parts should be allowed to fail/cheat, and the system should still work
  - Anonymity should still be enforced
  - Correct result should be obtained
- **Fairness** – No partial results should be disclosed before the end of the voting procedure
- Additionally
  - It should be easy to vote
  - Voting should be optional

Election Types

- **Yes/no** – Only two options
- **1-out-of-L** – Voters choose from one out of L options
- **K-out-of-L** voting – Voters choose K from L options
- **K-out-of-L ordered** voting – order the K choices
- **Write-in voting** – Freely chosen text strings

Building Blocks

- Building blocks we will use
  - *Chaum Mix*
  - *Blind Signatures*

  These you have seen before
  - ElGamal encryption and Homomorphic encryption
  - Zero-Knowledge Proofs
  - Secret Sharing and Threshold encryption
  - Commitment Schemes
- **Note:** See the crypto courses for more theoretical details – we just look at how they work and how to use them
  - You should get a feeling for how they work and fit together

ElGamal Encryption

- Asymmetric encryption – public/private key pair
- Based on discrete logarithm problem
  - Find *x* such that $y = g^x \bmod q$
- *x* is the *private key*, *y* is the *public key*, *g* and *q* are known
- **Encryption** of *m*:
  - Choose random *r*

$$E(m, r) = (a, b) = (g^r, m \cdot y^r)$$

- **Decryption**

$$\frac{b}{a^x} = \frac{m \cdot g^{xr}}{g^{xr}} = m \bmod q$$

Homomorphic Property

- Encryption scheme is homomorphic if

$$E(m_1) * E(m_2) = E(m_1 *' m_2)$$

for some operations $*$ and $*'$

- Homomorphic property of ElGamal encryption

$$E(m_1, r_1)E(m_2, r_2) = E(m_1 m_2, r_1 + r_2)$$

- since

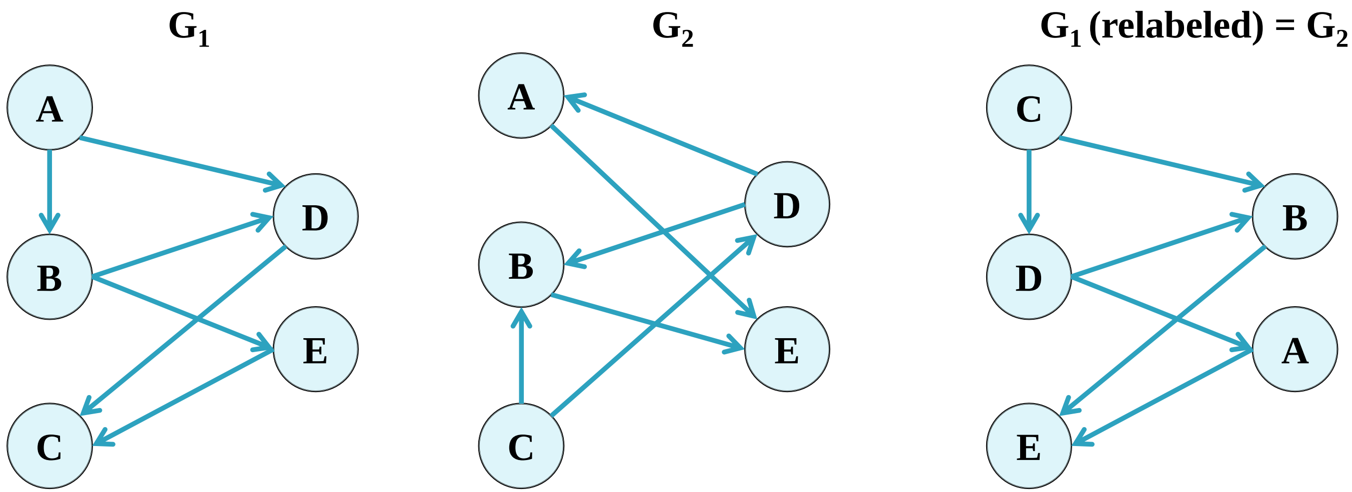
$$E(m_1, r_1)E(m_2, r_2) = (a_1 a_2, b_1 b_2)$$
$$= (g^{r_1+r_2}, m_1 m_2 y^{r_1+r_2}) = E(m_1 m_2, r_1 + r_2)$$

Zero-Knowledge Proofs

- Prove that you know a secret without revealing anything about the secret
  - Compare to the cut-and-choose method
- **Statement:** *"I know the number of leaves on this tree"*
- Can we prove this without revealing anything about the algorithm?
  - We are allowed to use interaction between Peggy (the prover) and Victor (the verifier)

Isomorphism Between Graphs

- Can we relabel the vertices such that two graphs are the same



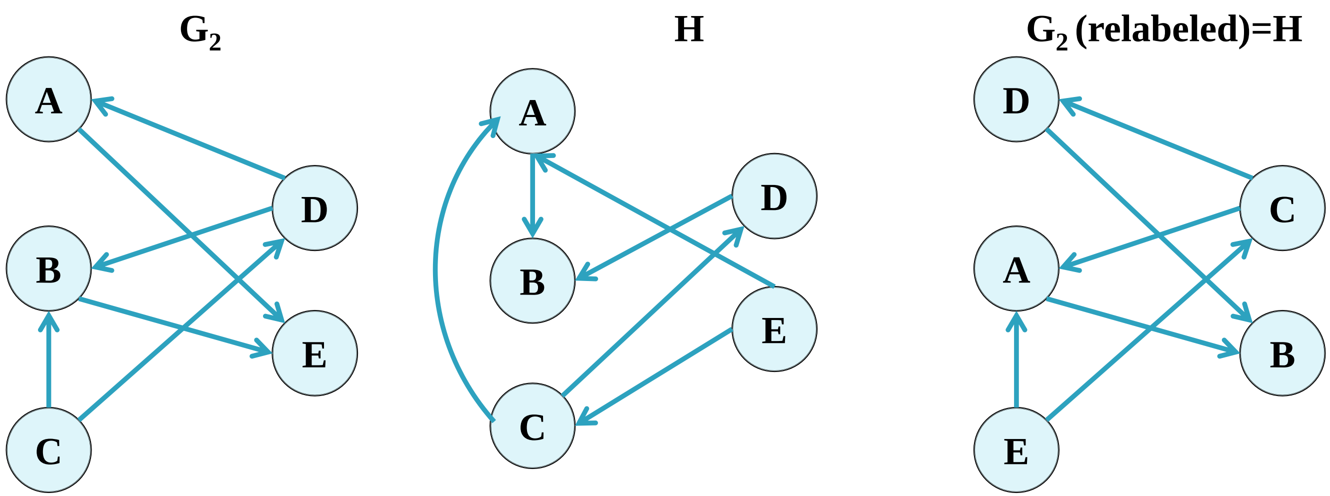
Isomorphism: $\phi : G_1 \to G_2$

A	B	C	D	E
C	D	E	B	A

- It is difficult to determine if two graphs are isomorphic
- It is very easy to verify an isomorphism between graphs

Proof of Knowing the Isomorphism

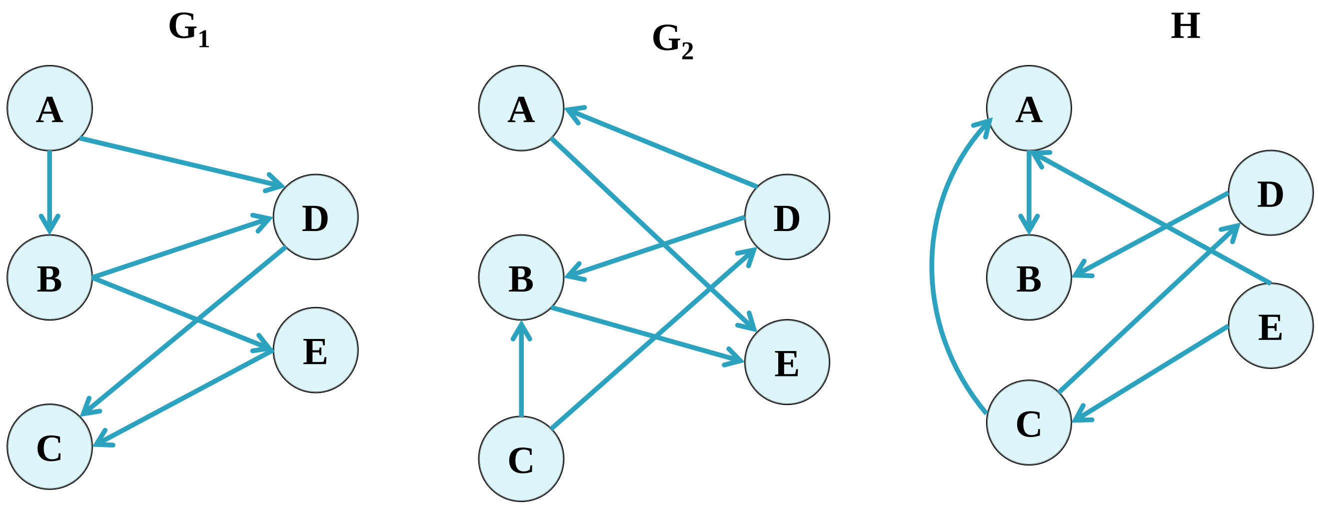
- Peggy: ”I know the isomorphism between the graphs”
- How can Victor verify this without Peggy revealing the isomorphism?
  - Peggy creates a graph H isomorphic to G_2



$$\gamma : G_2 \rightarrow H$$

A	B	C	D	E
D	A	E	C	B

Proof of Knowing the Isomorphism

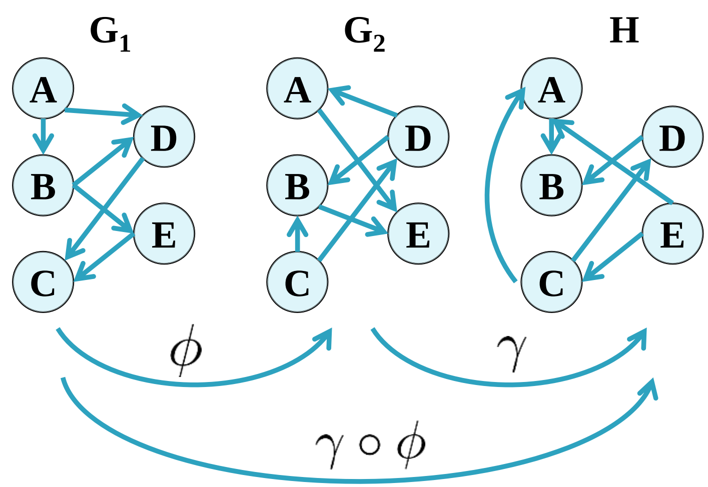


- Peggy now knows 3 isomorphisms
- Only one is secret
- **Idea:** Victor asks her to reveal γ or $\gamma \circ \phi$

$$\phi : G_1 \rightarrow G_2$$
$$\gamma : G_2 \rightarrow H$$
$$\gamma \circ \phi : G_1 \rightarrow H$$

Proof of Knowing the Isomorphism

- G_1 and G_2 are public

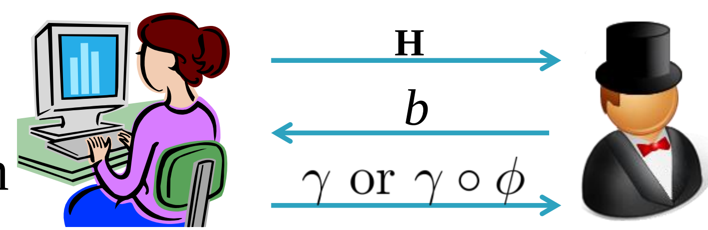


Protocol

1. Peggy sends H to Victor
2. Victor flips a coin and asks for isomorphism

$$G_b \rightarrow H$$

   - Heads: $b=1$
   - Tails: $b=2$
3. Peggy returns permutation (isomorphism)



Repeating the Protocol

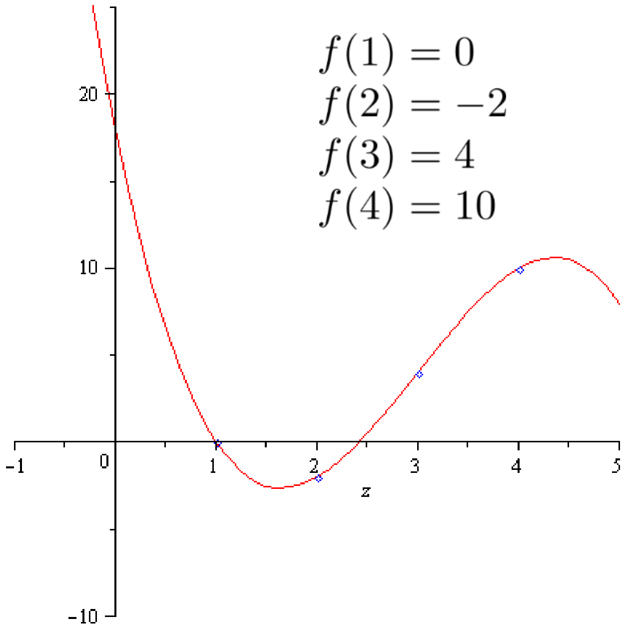
- Assume Peggy does NOT know isomorphism
  - Peggy could make H isomorphic to G_1 and hope that Victor asks for $\gamma \circ \phi$ $(b=1)$
  - Peggy could make H isomorphic to G_2 and hope that Victor asks for γ $(b=2)$
- So she fools Victor with probability 0.5
- Repeat protocol k times $\rightarrow$ Peggy can cheat with probability 2^{-k}
- Proof is **zero knowledge** if it is possible for Victor to **simulate** the communication
  - Produce a valid transcript of the communication between Peggy and Victor (without knowing the secret)

Secret Sharing

- Share a secret between several parties
- *(t,n)* threshold scheme
  - *n* parties get one share each
  - *t* need to cooperate to recover secret (*t-1* parties does not get any information about the secret)
- **Insight:** with *t* points on a polynomial of degree *t-1*, it is possible to recover the polynomial
  - Lagrange interpolation
  - Called Shamir secret sharing
- Use **trusted dealer** that constructs the polynomial *f(z)* and hands out shares
  - Secret *x* is given by $f(0)=x$

Lagrange Interpolation

- With *t* points on a curve, there is one polynomial of degree *t-1* that fits the curve



$$f(1) = 0$$
$$f(2) = -2$$
$$f(3) = 4$$
$$f(4) = 10$$

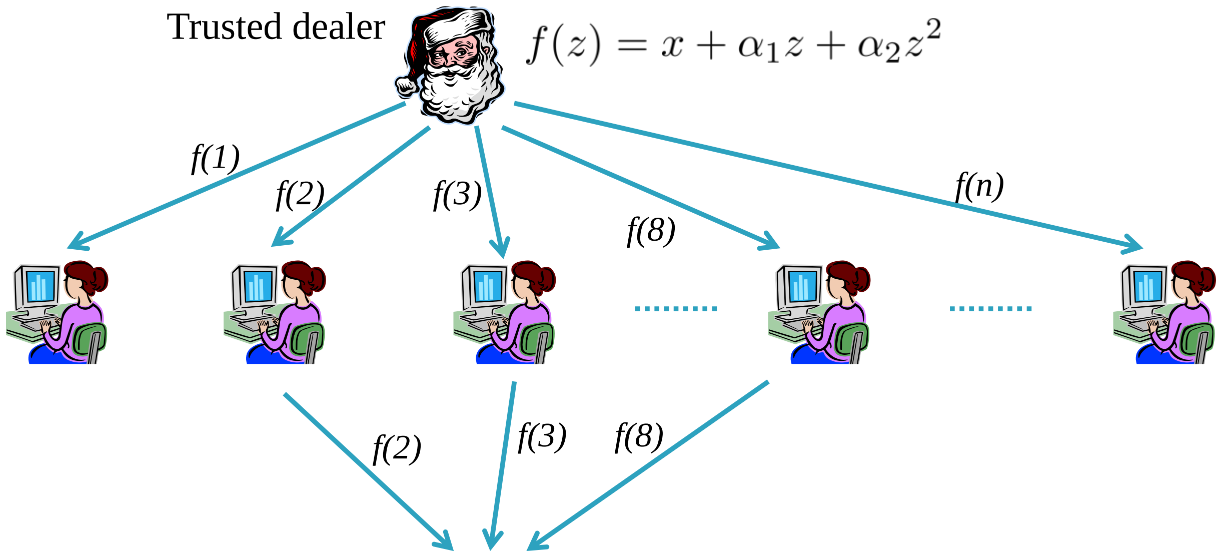
$$f(z) = \sum_{i=1}^{t} f_i(z)$$

$$f_i(z) = f(z_i) \prod_{j=1, j\neq i}^{t} \frac{z - z_j}{z_i - z_j}$$

If we only want *f(0)*, simplify to

$$f(0) = \sum_{i=1}^{t} f(z_i) \prod_{j=1, j\neq i}^{t} \frac{z_j}{z_j - z_i}$$

Example: (3,n) Threshold Scheme



$$f(z) = x + \alpha_1 z + \alpha_2 z^2$$

$$\Lambda = \{2, 3, 8\}$$

$$\begin{aligned} x &= f(0) = \sum_{i \in \Lambda} f(i) \prod_{j \in \Lambda \setminus \{i\}} \frac{j}{j - i} \\ &= f(2)\frac{3}{3-2}\frac{8}{8-2} + f(3)\frac{2}{2-3}\frac{8}{8-3} + f(8)\frac{2}{2-8}\frac{3}{3-8} \end{aligned}$$

Threshold Encryption

- Public key setting
- Let all participants form a public/private key pair
- Form one public key from the individual public keys
- Require at least *t* participants in order to reconstruct the private key
  - No trusted dealer

$y_1 = g^{x_1} \bmod q$

$y_2 = g^{x_2} \bmod q$

$y_n = g^{x_n} \bmod q$

- Public key $y_1 y_2 \cdots y_n = g^{x_1 + x_2 + \ldots + x_n} \bmod q$
- Private key $x_1 + x_2 + \ldots + x_n$
- Let *t* participants recover $x_1 + x_2 + \ldots + x_n$

Secret Sharing of Private Values

- Each participant acts as trusted dealer for shares of her own private key
- Everyone constructs

$$f_i(z) = x_i + \alpha_{i,1} z + \ldots + \alpha_{i,t-1} z^{t-1}$$

- Participant *j* gets the share $f_i(j)$
- Sum of all polynomials

$$\begin{aligned} f(z) &= f_1(z) + f_2(z) + \ldots + f_n(z) \\ &= x + \sum_{i=1}^{n} \alpha_{i,1} z + \sum_{i=1}^{n} \alpha_{i,2} z^2 + \ldots + \sum_{i=1}^{n} \alpha_{i,t-1} z^{t-1} \end{aligned}$$

- Participant *i* can compute one point on this curve

$$f(i) = f_1(i) + f_2(i) + \ldots + f_n(i).$$

- Now *x* can be recovered with *t* such points

Using a Bulletin Board

- Many schemes uses (or imagines) a bulletin board
- **Everyone can read** everything on the board
- Each user has his own section of the board he can write to
  - Can not write to anything else
  - Only append rights are given – not possible to make changes
- Often modelled as a broadcast channel with memory

- It can be used to provide *universal verifiability*

Making an Electronic Voting Scheme

- The trick is to combine **Privacy** and **Universal verifiability**
- Two main strategies
  - The vote is posted on the bulletin board in clear text, but the person casting the vote is anonymous
  - The vote is posted on the bulletin board in encrypted form, and the person is not anonymous
- Three main methods
  - Mix networks
  - Blind signatures
  - Homomorphic encryption

Using a Mix Network

- Mixes enable anonymity – voting requires anonymity
- Vote using a pseudonym – a public key (PK)

Registration phase

$$K_n(R_n, K_{n-1}(\ldots, K_2(R_2, K_1(R_1, PK))\ldots)).$$

- First Mix can check voter eligibility
- Last Mix will output list of pseudonyms of eligible voters
  - Write to bulletin board
- Now everyone can see that they are registered voters
  - If not – complain

Bulletin Board

$$\begin{array}{l} PK_1 \\ PK_2 \\ PK_3 \\ \vdots \\ PK_n \end{array}$$

Mix Network, Voting Phase

Voting phase

- Public key, vote and signature on vote is sent through the Mix network

$$K_n(R_n, K_{n-1}(\ldots, K_2(R_2, K_1(R_1, PK, V, \sigma(V)))\ldots))$$

- Last Mix posts public key together with each vote
- Anyone is able to count the votes and check the result
- There is no robustness
  - If one mix behave erronously, there will be errors
- There is no universal verifiabilty
  - Users can check their own vote for correctness, but not other votes
- If there is an error in a voter's vote, all other votes are disclosed → no fairness

Bulletin Board

PK_1	V_1	$\sigma_{SK_1}(V_1)$
PK_2	V_2	$\sigma_{SK_2}(V_2)$
PK_3	V_3	$\sigma_{SK_3}(V_3)$
$\vdots$	$\vdots$	$\vdots$
PK_n	V_n	$\sigma_{SK_n}(V_n)$

Mix Networks

- Easy to support many different types of systems since votes are all in clear text
  - Yes/no, 1-out-of-L, K-out-of-L, K-out-of-L ordered, Write-in, etc
- Requires anonymous channel
- Voters work can be made independent of the number of mixes, though this is not seen explicitly in the examples given here
  - Use El-Gamal and re-encryption
- Universal verifiability is possible
  - Add all message steps to bulletin board
  - Let Mixes prove their behaviour to everyone

Improvements – Rough Ideas

- Voters post encrypted votes on bulletin board
- Separate permutation and decryption phases and prove correctness

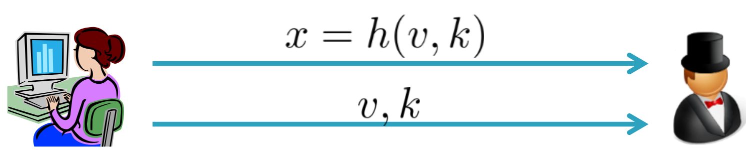
1. Each server randomizes and permutes the encrypted votes
2. Prove correctness of previous step, either individually for each mix or all mixes together (with erroneous Mixes being identified)
   - Prove that they know randomness and permutation that maps input to output
   - Proof based on e.g., zero-knowledge or cut-and-choose
3. *t* out of *n* servers decrypts the message (threshold decryption)
4. Prove correctness of previous step (again with erroneous Mixes being identified)
   - If necessary – pick another set of *t* Mixes for decryption

Using Blind Signatures

- Separate Administrator and Counter
  - Administrator identifies voter
  - Counter collects votes that have been blindly signed by Administrator
  - Privacy should hold even if they cooperate
  - Still – requires an anonymous channel

Commitment Schemes

- Alice wants to commit to a value to Bob
- Two steps
  - Commitment stage – Alice sends commitment to Bob
  - Revealing stage – Alice reveals the value commited to
- Most straightforward way is to use a hash and a random value *k*



- Properties
  - Binding – Sender can not change her mind after committing to the value
  - Concealing – Receiver can not determine value of *v* before revealing
- Binding and concealing can be information theoretic or computational

Voting Protocol Using Blind Signatures

Voter registration phase

- Voter V_i makes a commitment to her vote $x_i = h(v_i, k_i)$
- Vote is blinded and sent to Administrator together with a signature $e_i = \chi(x_i, r_i), \quad s_i = \sigma_{V_i}(e_i)$
- Administrator checks voter eligibility and signs blinded commitment $d_i = \sigma_A(e_i)$
- Administrator publishes list


Voter

$e_i, s_i \rightarrow$

$\leftarrow d_i$

Administrator


$\langle ID_i, e_i, s_i \rangle \rightarrow$

Bulletin Board

ID_1	e_1	s_1
ID_2	e_2	s_2
ID_3	e_3	s_3
$\vdots$	$\vdots$	$\vdots$
ID_n	e_n	s_n

- Everyone can see list of accepted voters together with blinded commitment and signature

Voting Protocol Using Blind Signatures

Voting phase

- Voter extracts Administrator's signature on the commitment and sends this anonymously together with commitment to Counter
- Counter verifies signature and writes list to bulletin board


Voter

$$x_i, y_i = \sigma_A(x_i) \longrightarrow$$

Counter


Bulletin Board

1	x_1	$y_1 = \sigma_A(x_1)$
2	x_2	$y_2 = \sigma_A(x_2)$
3	x_3	$y_3 = \sigma_A(x_3)$
$\vdots$	$\vdots$	$\vdots$
n	x_n	$y_n = \sigma_A(x_n)$

- Everyone can compare the number of rows in the two lists
- Each voter checks that her commitment is included
  - If not – reveal *r* (which together with *x* gives *e*), but not the actual vote

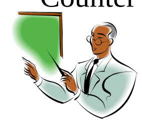
Voting Protocol Using Blind Signatures

Tallying phase

- Voter V_i sends (l_i,k_i) anonymously to Counter
- Counter adds v_i and k_i to the bulletin board


Voter

$\xrightarrow{v_i, k_i}$

Counter


Bulletin Board

1	x_1	y_1	v_1	k_1
2	x_2	y_2	v_2	k_2
3	x_3	y_3	v_3	k_3
$\vdots$	$\vdots$	$\vdots$		
n	x_n	y_n	v_n	k_n

$$x_i = h(v_i, k_i)$$

- Everyone can check the result
- Everyone could check that their vote was counted before any votes were revealed
- Administrator can not link specific vote to specific signature
  - Privacy maintained
- **Main problem:** Universal verifiability is not possible

Homomorphic Encryption Based Voting Schemes

- Compute result without opening individual votes
  - Users do not have to be anonymous since vote is encrypted
- Homomorphic property of ElGamal encryption

$$E(m_1, r_1)E(m_2, r_2) = E(m_1 m_2, r_1 + r_2)$$

- Not very useful – We want the sum of votes, not the product
- Modified ElGamal

$$E(m, r) = (a, b) = (g^r, w^m y^r)$$

- Homomorphic property

$$E(m_1, r_1)E(m_2, r_2) = (a_1 a_2, b_1 b_2) = (g^{r_1+r_2}, w^{m_1+m_2} y^{r_1+r_2}) = E(m_1+m_2, r_1+r_2)$$

which is exactly what we want

- If sum of m_i is moderate we can compute the discrete log

Steps in Voting Scheme

1. A user encrypts the vote using a homomorphic threshold scheme. We use El Gamal encryption here. The encrypted vote is published on a bulletin board so everyone can see who has voted.
2. The voter proves that the vote is valid.
3. Multiply encrypted votes
4. A set of authorities cooperate to decrypt the sum or product
5. Everyone can verify that the product of the encrypted votes is indeed a valid encryption of the final result. This gives universal verifiability.

More Threshold Encryption

- If *t* authorities are needed to decrypt, then *t* need to be malicious in order to break privacy
- Recall secret sharing scheme

$y_1 = g^{x_1} \bmod q$ $y_2 = g^{x_2} \bmod q$ .... $y_n = g^{x_n} \bmod q$

- Public key $y_1 y_2 \cdots y_n = g^{x_1+x_2+\ldots+x_n} \bmod q$
- Private key $x_1 + x_2 + \ldots + x_n$
- Add the following (amazing) properties:
  - A message will be decrypted without anyone learning the private key
  - Authorities will prove that they are behaving correctly when they participate in the decryption

More Threshold Encryption

- Each authority A_i computes a polynomial in order to share x_i

$$f_i(z) = x_i + \alpha_{i,1} z + \ldots + \alpha_{i,t-1} z^{t-1}$$

- Each authority will receive $f_i(j)$
- Each authority sums his shares

$$f(i) = f_1(i) + f_2(i) + \ldots + f_n(i).$$

…and gets a point on the curve

$$f(z) = f_1(z) + f_2(z) + \ldots + f_n(z)$$

We call this point s_i for authority A_i. Commit to s_i by publishing

$$h_i = g^{s_i} \bmod q$$

Decrypting the El Gamal Threshold Scheme

- Decrypt $(a, b) = (g^r, my^r)$
- Every authority publishes

$$h_i = g^{s_i} \bmod q$$

$$u_i = a^{s_i} \bmod q$$

- Λ is a set of *t* authorities
- Now, *m* is decrypted as

$$m = \frac{b}{\prod_{i\in\Lambda} u_i^{\lambda_{i,\Lambda}}} \left(= \frac{b}{g^{r\sum_{i\in\Lambda} s_i\lambda_{i,\Lambda}}} = \frac{b}{y^r} \right)$$

where

$$\lambda_{i,\Lambda} = \prod_{j\in\Lambda\setminus\{i\}} \frac{j}{j-i}$$

- Authorities prove in zero-knowledge that $\log_g h_i = \log_a u_i$.

Proving Correct Behaviour

- Authorities must prove that

$$\log_g h_i = \log_a u_i.$$

Prover		**Verifier**
Proves that $(h, u) = (g^s, a^s)$		
Pick $w \in \mathbb{Z}_q$ and compute $(a', b') = (g^w, a^w)$	$\xrightarrow{(a', b')}$	
	$\xleftarrow{c}$	Pick random $c \in \mathbb{Z}_q$
Compute $r = w + sc$	$\xrightarrow{r}$	
		Check that $g^r = a'h^c$ and $a^r = b'u^c$

Applied to Electronic Voting

1. Each voter V_i encrypts $v_i = -1$ or $v_i = 1$
$$E(v_i, r_i) = (a_i, b_i) = (g^{r_i}, w^{v_i} y^{r_i}).$$
2. Voter proves that vote is actually $v_i = -1$ or $v_i = 1$
3. Encrypted vote and proof written to bulletin board
4. When everyone has voted, multiply all encryptions
$$\prod_{i=1}^{m} E(v_i, r_i) = (\prod_{i=1}^{m} a_i, \prod_{i=1}^{m} b_i) = (g^{\sum_{i=1}^{m} r_i}, w^{\sum_{i=1}^{m} v_i} y^{\sum_{i=1}^{m} r_i}) = E(w^{\sum_{i=1}^{m} v_i}, \sum_{i=1}^{m} r_i).$$
5. Authorities publish $u_i = a^{s_i} \bmod q$ and proves that $\log_g h_i = \log_a u_i$.
6. Decrypt using *t* honest authorities (those that pass the proof)
7. Recover result by solving discrete logarithm
   - Possible if number of voters is not too many

Properties of Voting Scheme

- Works well for yes/no voting
  - Some other types work as well
- Universal verifiability – everyone can check result
- Robustness – only *t* authorites need to be honest
- Not much job for authorities
  - More work in Mix networks
- Need to solve discrete logarithm
  - Other encryption schemes can be used, e.g., Paillier encryption
- Zero knowledge proof for vote validity is needed
  - Not needed in Mix networks