A voting system case study:Pret-A-Voter

Panagiotis Grontas

$\mu\Pi\lambda\forall$ - CoReLab Crypto Group

26/06/2013

Verifiable Elections With Receipts I

Cryptographic Voting Receipts

- Assure Voters that their vote is counted
- While maintaining ballot secrecy and avoiding coercion

Advantages

- Mathematically ensure election integrity
- Allow software independence [Riv08]
  - Avoid trust on proprietary hardware
  - Detect errors early (at the polling station)
  - Rerun the elections without software

Verifiable Elections With Receipts II

Software independence

A voting system is software independent if an undetected change or error in its software cannot cause an undetectable change or error in the election outcome.

How:

- Paper trail
- Cryptography

Voting with receipts

- Input choices on the touch screen of the voting machine
- Printer prints the receipt
- Voter checks the printout
- Accept or Change Vote
- Mechanism for ballot secrecy
- Receipt
- Mechanism for receipt secrecy
- Anonymization
- Tallying
- Verification: Web Bulletin Board to enter receipt SN and verify that it is included in the tally

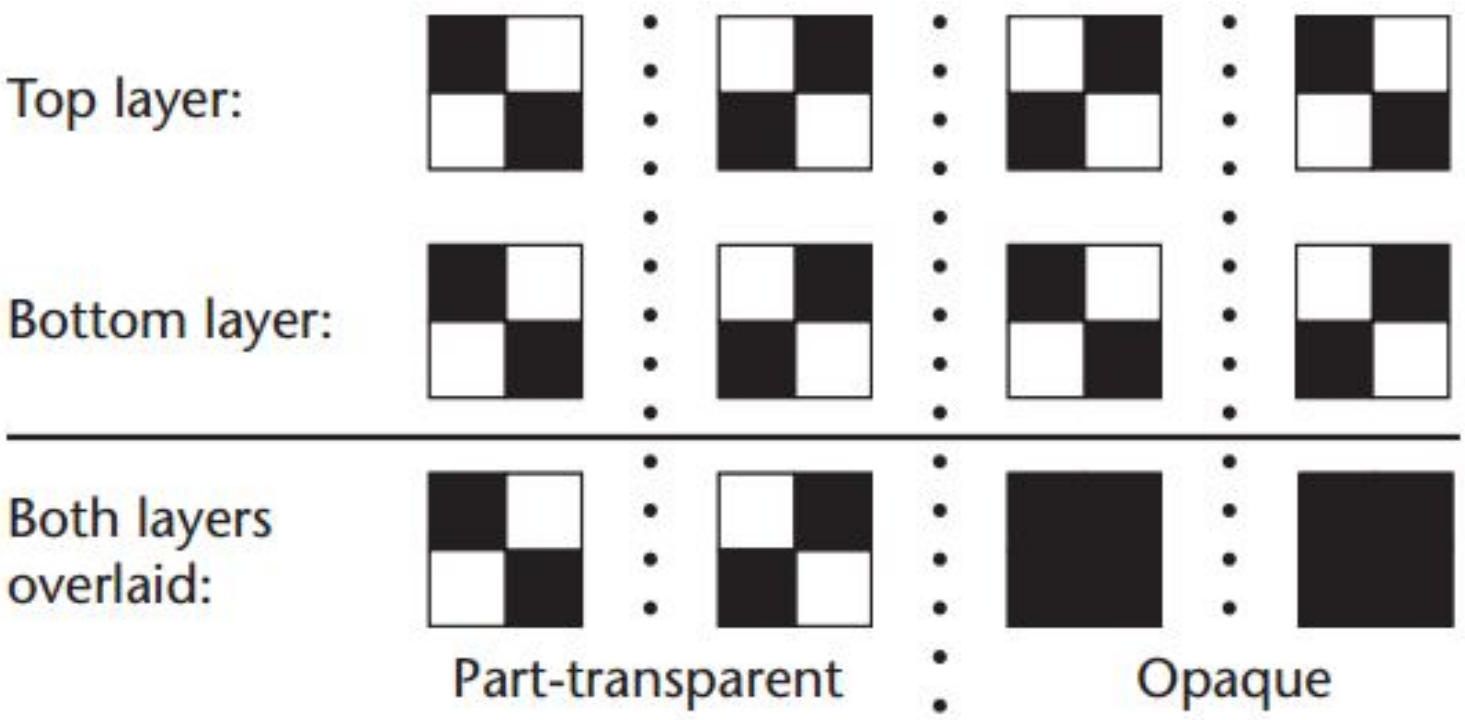
Visual Cryptography

- Encrypt written material in a **perfectly secure** way
- Decryption is done by human visual system
- Ciphertext and secret key are two pages of papers
- Ciphertext can be transmitted by fax or published
- Secret key is printed on transparent paper
- Each one on its own is indistinguishable from random noise
- When placed on top of each other we have decryption
- A visual one time pad
  - Each page of cipher text is decrypted by its own transparency
- Can be extended to k of n threshold secret sharing scheme [NS95]

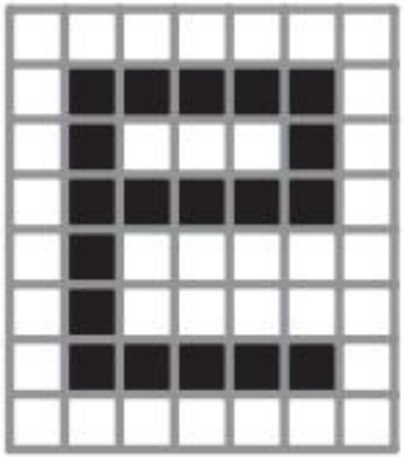
Receipts with Visual Cryptography - [Cha04]

- Receipts consist of two layers laminated together
- The printer prints the selection of the voter but does not conclude the job
- Voter selects to keep top or bottom layer and then the printing concludes
- Machine keeps the same layer in digital form
- Neither layer is readable on its own, but both encode the vote
- The other layer is destroyed by the voting personnel and the voting booth (securely?)

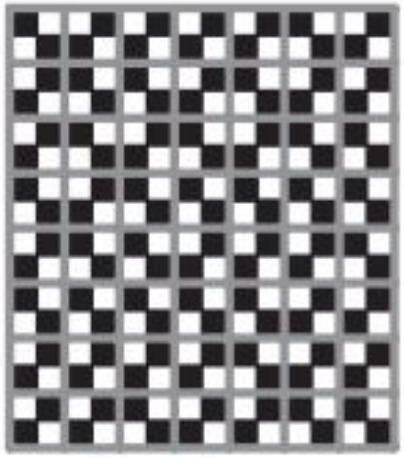
Encoding the vote with visual cryptography I



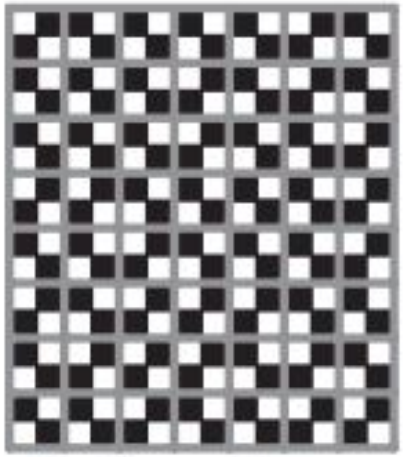
Encoding the vote with visual cryptography II



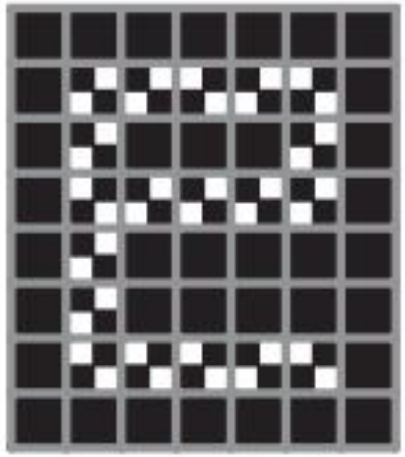
Newspaper



Top layer



Bottom layer



Laminated

PretAVoter - Introduction

- End To End Verifiable Voting System
- A design of ballots for electronic voting
- Creator: Peter Ryan (University of Luxemburg) (2004)
- Based on a previous cryptographic scheme by Chaum (2004)
- Influenced a subsequent cryptographic scheme by Chaum
- First implementation by University of Surrey (2007)
- Still in active development

PretAVoter - Characteristics

- Integrity
- Voter Privacy
- Receipt Freeness
- Voter Verifiability
- Voter Friendly
- Modular: Can implement many cryptographic voting protocols (encryption, zero knowledge etc.)
- Versatile: Can be used with many social choice functions
- Requires supervised voting infrastructure

Assumptions

- Voter registration and authentication is outside the scope of the system
- The ballot forms are properly managed (remain confidential, cannot be forged) between their creation and used
- Voting occurs in a voting booth with the traditional privacy characteristics
- There exists a tamper - free public bulletin board that can be used for verification

The PretAVoter Ballot I

Main idea

The vote is recorded on a different frame of reference for each ballot.
As a result the system **encrypts the ballot not the vote**

- The ballot consists of two columns: one for the candidate list and one for the voter selection.
- The candidate list is random for each ballot
- The voter marks or ranks the candidates on his/her ballot
- The candidate list **must** be disposed
- The actual vote is the mark or rank that is recorded on the vote column
- No encryption is needed, since the vote is useless without the candidate list
- The vote column is used as a receipt for voter verifiability

The PretAVoter Ballot II

- Since the vote order is different, a source of voter bias is removed but compliance might be affected

Candidates	Your vote
Obelix	
Idefix	
Asterix	
Panoramix	
	$7rJ94K$
Destroy	Retain

Your Vote
X
$7rJ94K$
Retain

Candidates	Your vote
Asterix	
Idefix	
Panoramix	
Obelix	
	$N5077t3$
Destroy	Retain

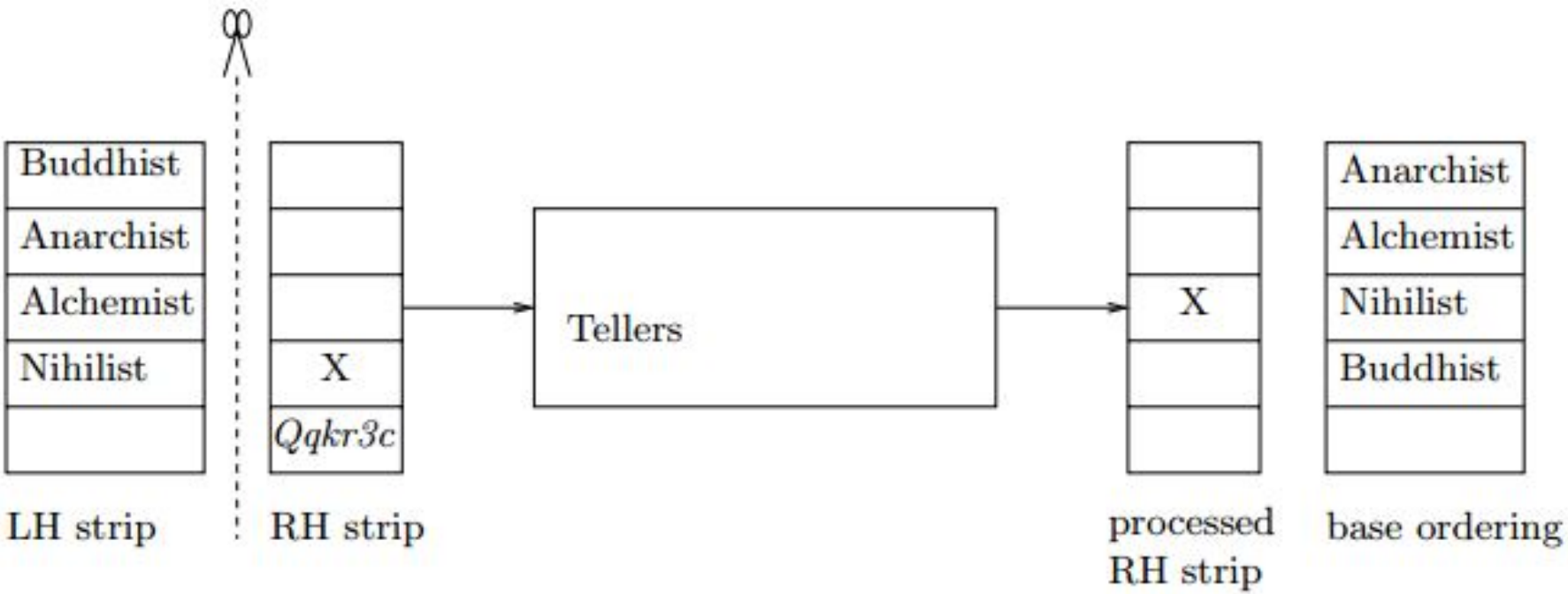
- There is a base ordering of a candidates
- The ordering on each ballot is a permutation of the base

The PretAVoter Ballot III

- The difference (offset) is encrypted and printed on each ballot
- The encrypted value is called onion, referring to the processing of the ballot in the first version of the systems
- The voting equipment records the index/ranking and transmits it along with the encrypted onion
- Only the voter knows the order of the candidates that he voted on
- **Extra bonus:** side channel attacks are side stepped
  - No encryption takes place at the booth (the onion is created in advance)
  - The vote without the candidates is useless

Vote Processing I

1. Mixing
2. Decrypting
3. Tallying



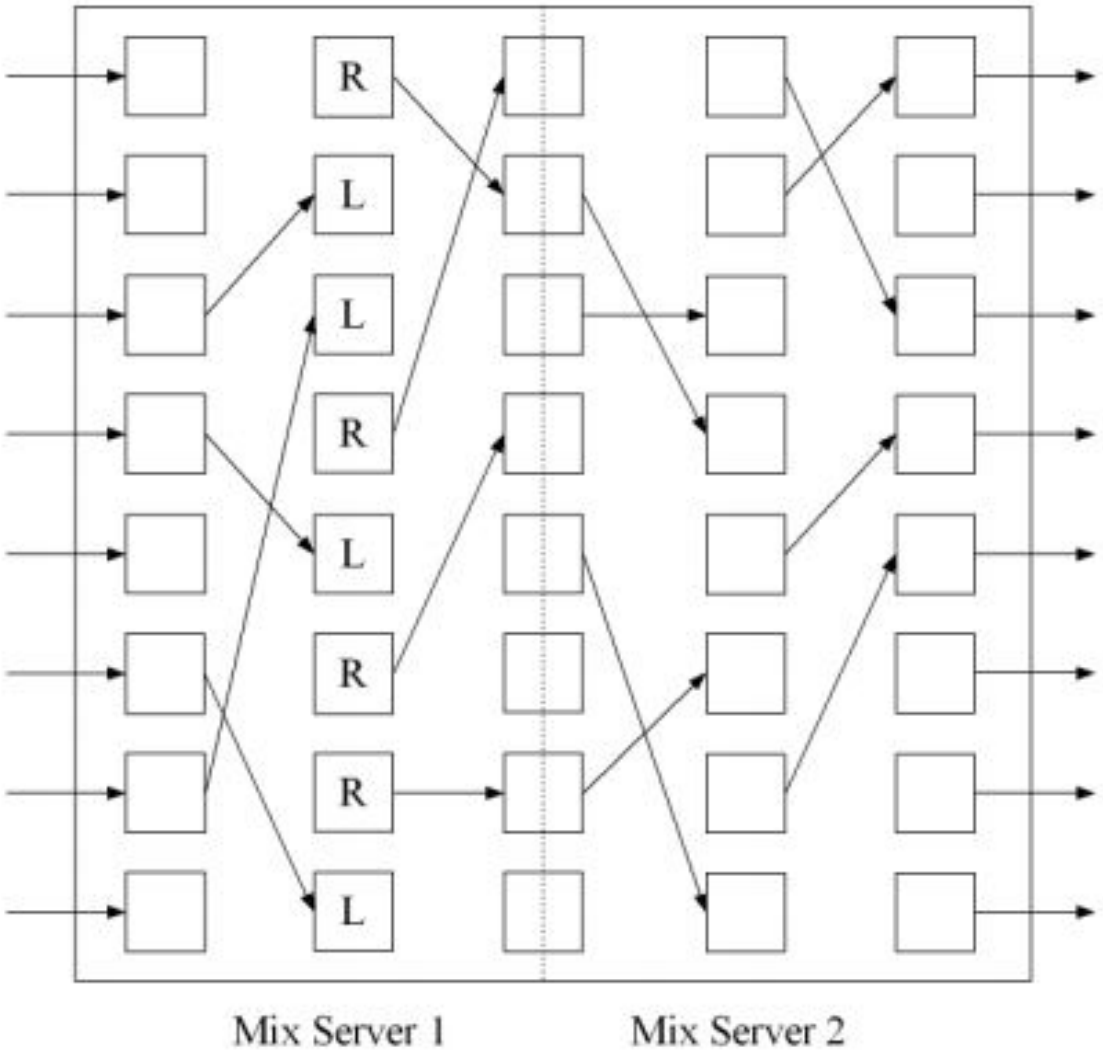
Decryption Mixnet I

- n voters
- m candidates
- Each mix server performs 2 mixes
  - k mix servers
  - $2k$ key pairs
- Ballot generation - Layered Generation
  - $2k$ random values $\{g_{i,j}\}_{0,0}^{n-1,2k-1}$ - germs - for each voter
  - the germs will be used for
    - **randomised encryption**
    - Onion for ballot i: $D_{i,2k} = Enc(g_{i,2k-1}, Enc(\cdots Enc(g_{i1}, Enc(g_{i0}, D_{i0})), \cdots))$ where D_{i0} random value for initialisation
    - **offset calculation**
    - $\theta_i = \sum_{j=0}^{2k-1} d_j \mod m$ where $d_j = hash(g_{ij})$

Decryption Mixnet II

- Vote i is a pair $(r_i, D_{i,2k})$ where r_i is the index of the marked candidate
- Mix server operation
  - Input $L_{2j} = \{r_i, D_{i,2j}\}_{i=0,j=k-1}^{n-1,0}$
  - Decrypt the onion with the private key (g_ij, D_{ij})
  - Apply the hash to the germ value of the onion $d_{ij} = hash(g_{ij})$
  - Calculate new shift $r_{ij} = r_{i,j-1} - d_{ij}$
  - Sort the list by the value of the onions
  - Audit with RPC

Decryption Mixnet III



Threats

- Confidentiality of ballots, before elections. There is a need to trust the authority and the intermediates
- Chain Voting: Coercion by smuggling a ballot, pre-voting, and require a replacement ballot
- Kleptographic Channels: Choose special values of cryptographic variables in order to leak information to a colluding party
  - Tagging attack in mixnets

Common Theme: Too much trust in a single authority

Solution: Distributed Generation of Ballots

Distributed Generation Of Ballots I

- The ballot forms will be generated by l entities called *clerks*
- Each clerk contributes to the generation of a common encrypted seed
- The candidate list is constructed from the seed
- To reveal the seed values all clerks must be corrupted
- Encryption using exponential ElGamal (also works for RSA)
- Tellers conduct tallying using a reencryption mixnet

Distributed Generation Of Ballots II

Operation

- Generate 2 encrypted *'onions'*:
- $onion_L$
  - will create the random candidate list
  - will be read (decrypted) by the voting machine in order to display the candidates to the voter
- $onion_R$
  - will be used for voter verifiability
  - and reconstruction of the candidate list

Distributed Generation Of Ballots III

Parameters

- p, q safe primes
- α, γ generators of $\mathbb{Z}_q^*$
- β_T the public key of the tellers
- β_R the public key of the vote recording machines

Clerk C_0

- Generate a batch of initial seeds s_i^0 - one for each ballot
- Create encryption with randomness x_i^0, y_i^0 for
- voting machine $E_R(\gamma^{-s_i^0}, x_i^0) = (\alpha^{x_i^0}, \beta_R^{x_i^0} \cdot \gamma^{-s_i^0})$
- teller $E_T(\gamma^{-s_i^0}, y_i^0) = (\alpha^{y_i^0}, \beta_T^{y_i^0} \cdot \gamma^{-s_i^0})$

Distributed Generation Of Ballots IV

Clerk C_l

- Receive batch of onions from previous clerk
- Generate fresh randomness
- Perform reencryption and shuffle

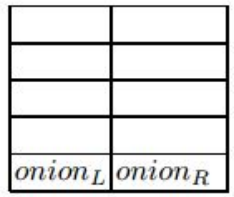
Final onions

- $onion_L = E_R(\gamma^{-\sum_i s_i}, \sum_i x_i)$
- $onion_R = E_T(\gamma^{-\sum_i s_i}, \sum_i y_i)$

Correct operation $\Rightarrow$ *the 'messages' match.*

Voting with distributed ballots I

- The onions are stored in encrypted form and are presented to the voter on the ballot form



- Device reads and decrypts the left hand onion s is revealed
- The candidate permutation is reconstructed as a function of s
- The device should not see the right onion

Voting with distributed ballots II

Voting

- The voter now faces a classical ballot
- Marks candidate index
- Embed selected candidate index into seed by reencryption
  $vote = E_T(\gamma^{r-\sum_i s_i}, \sum_i y_i)$
- Mixing proceed as usual
- Decryption: plaintexts are of the form $\gamma^{r-\sum_i s_i}$
- Solve discrete logarithm
- Select seeds so that the exponent is not too large and the search space is bounded
- **Solution**: Use Paillier Cryptosystem

Overview of STV Voting I

Objective:

Reduce vote waste, for candidates that cannot win the election

- Each voter gets a single vote, but 'this vote' can be used by many candidates
- When a candidate is eliminated, 'his' votes are transferred
- Each voter ranks candidates
- Ranking selects the candidates to receive the votes in case favourites are eliminated
- **Tallying**
  - Partition ballots according to the first choice of the voter
  - The candidate with the fewest votes is eliminated
  - His votes are transferred according to the second choices
  - Repeat until there is only one candidate

Overview of STV Voting II

Remarks

- **Quotas:** The number of votes that assure election
- If a candidate has reached the quota, he cannot be eliminated
- But the surplus of votes has to be transferred, as well
- The voter should not provide a complete ranking of the candidates.
- A ballot with all the candidates eliminated is discarded

Overview of STV Voting III

Useful Characteristics

- **Unused Preferences:** Some of the preferences might not be used, since the winner might get declared earlier.
- **Transfer history** is irrelevant. We are not interested for the previous candidates since they are eliminated
- **Lazy evaluation** can be applied. We can work on the candidates one by one.

Difficulties

- **The italian attack** Many preferences introduct a covert channel, as lower ranked preferences can be used to identify the voter

Incorporating STV into PretAVoter [Hea07] I

Attempt 1

- Encode a partial ranking in the vote
- Keep the cyclic shift
- But this might leak too much information because it preserves the relations found in the canonical ordering
- An example:

People's Front of Judea 1	
Judean People's Front 1	
People's Front of Judea 2	
Judean People's Front 2	
Judean People's Front 3	
People's Front of Judea 3	

	1
	6
	3
	2
	4
	5

- We must embed a complete permutation

Incorporating STV into PretAVoter [Hea07] II

Attempt 2 - ElGamal

- Encode a permutation in the ElGamal exponent in a way that addition/multiplication yields permutation composition
- Cannot be done

Attempt 3 - RSA onions

- Each layer encodes a permutation that can be recovered and applied
- **Problems**
  - It is evident how many choices were made
  - This can be used to partition ballots and leak information
  - No random padding can be added, since it will affect the election results
  - The candidates that were not ranked can be used as a covert channel (Italian attack)

Incorporating STV into PretAVoter [Hea07] III

Solution:Multi valued reencryption mixes

- Encode the vote as a sequence of cryptograms $\{c_1, \cdots, c_k\}$ that decrypts to a candidate index
- The order of the cryptograms reflects the ranking of the candidates by the voter, ie c_1 is the encryption of the index of the favourite candidate
- A dummy candidate 'STOP' id included that marks the end of the user ranking
- A complete ranking is given by using a random permutation of the remaining candidates
- **Lazy Tallying:** First decrypt the heads
- First preferences are counted
- If a candidate is eliminated then the head is appended at the end, an anonymising mixnet is run and the heads are again decrypted
- Stop when a winner (a candidate with more than half the votes) is found

STV with PretAVoter:An example I

Ballot Form

- Each ballot form contains a signed hash
- Hash serves as a key to lookup $k+1$ onions
  - The first k correspond to the candidates $\{i, z\}_{PK_T}, i = 1 \cdots k$ and are presented in random order
  - The last is the encryption of the stop candidate $\{0, z\}_{PK_T}$

Initial Ballot Form
Left Side: Registers
Right Side: Tellers
Encryption of candidate indexes

$[\text{Alice}]_{PK_R}$	$[\text{Alice}]_{PK_T}$
$[\text{Bob}]_{PK_R}$	$[\text{Bob}]_{PK_T}$
$[\text{Charlie}]_{PK_R}$	$[\text{Charlie}]_{PK_T}$
$[\text{Duncan}]_{PK_R}$	$[\text{Duncan}]_{PK_T}$
$[\text{Esther}]_{PK_R}$	$[\text{Esther}]_{PK_T}$
	$[\text{STOP}]_{PK}$

STV with PretAVoter:An example II

- Distributed Ballot Generation

$[Charlie]_{PK_R}$	$[Charlie]_{PK_T}$
$[Bob]_{PK_R}$	$[Bob]_{PK_T}$
$[Esther]_{PK_R}$	$[Esther]_{PK_T}$
$[Alice]_{PK_R}$	$[Alice]_{PK_T}$
$[Duncan]_{PK_R}$	$[Duncan]_{PK_T}$
	$[STOP]_{PK_T}$

Reordered Ballot Form

Dummy Position Candidate Unchanged

Only the registrars can read the candidate list on the left side

Only the tellers can read the onions on the right side

- Each ballot can be printed by a subset of registrars after being retrieved from a bulletin' board

STV with PretAVoter:An example III

Voting

Voter sees the decrypted ballot form, without the 'STOP' candidate and ranks the candidates (partially)

Charlie	
Bob	3
Esther	1
Alice	
Duncan	2
STOP	
	w4JKz

Booth sends the ranking and the hash to the bulletin board

STV with PretAVoter:An example IV

PostVoting Bulletin Board Actions

- First Available ranking is assigned to 'STOP' candidate
- Rest of empty voters are filled from top to bottom
- Use the hash to retrieve candidate list
- Sort Based on ranking

5
3
1
6
2
4
w4JKz

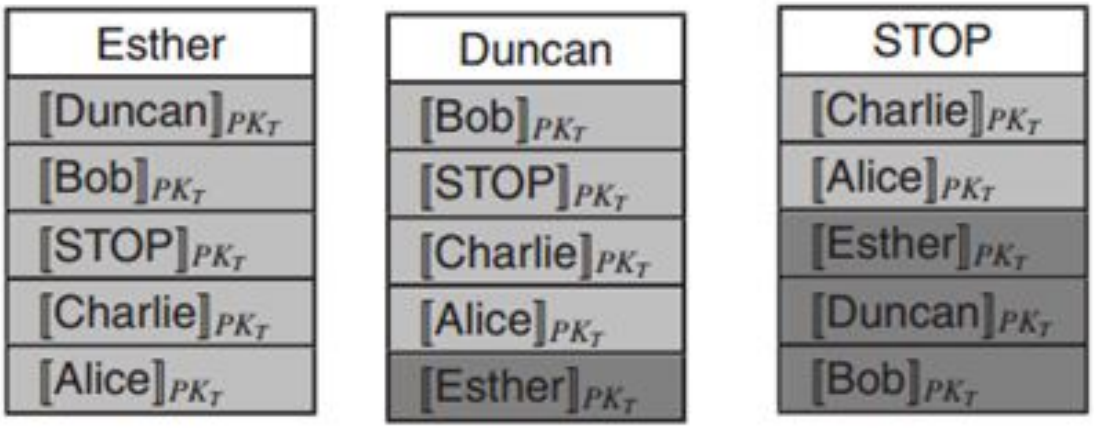
$[\![\text{Charlie}]\!]_{PK_T}$
$[\![\text{Bob}]\!]_{PK_T}$
$[\![\text{Esther}]\!]_{PK_T}$
$[\![\text{Alice}]\!]_{PK_T}$
$[\![\text{Duncan}]\!]_{PK_T}$
$[\![\text{STOP}]\!]_{PK_T}$
w4JKz

$[\![\text{Esther}]\!]_{PK_T}$
$[\![\text{Duncan}]\!]_{PK_T}$
$[\![\text{Bob}]\!]_{PK_T}$
$[\![\text{STOP}]\!]_{PK_T}$
$[\![\text{Charlie}]\!]_{PK_T}$
$[\![\text{Alice}]\!]_{PK_T}$
w4JKz

STV with PretAVoter:An example V

Anonymising and tallying

- Apply STV
- Decrypt candidates one at a time, until stop is reached
- After each decryption, another round of anonymisation occurs



Condorcet Voting

- The voter provides a full ranking of the candidates
- All candidates engage in pairwise comparisons (duels)
- The winner of each dual is the candidate that is preferred by the majority of the voters

An example - duel between the candidates c_A and c_B

For each ballot b do

if c_A is ranked higher than c_B in b then

$w_A \leftarrow w_A + 1$

else

$w_B \leftarrow w_B + 1$

$duel - winner \leftarrow$ candidate with $w = max(w_A, w_B)$

- The winner of the election is the one that wins **all** the duels
- There might not be a winner, so a completion method is used (but this is a backend issue)

Ballots for Condorcet Voting [CM05]

- For c candidates we have $\frac{c(c-1)}{2}$ $\dashrightarrow$ $O(c^2)$ ballots
- Each ballot is a yes/no choice for candidates i, j
- A yes choice indicates preference for candidate i over candidate j
- Each yes/no ballot has its own onion $< i, j >$
- Layered construction of onions for use in decryption mixnet
- Each voter submit $c \times c$ votes with the preferences and the onions
- The vote is run through a decryption mixnet for anonymisation
- **Remark**: There can be no correlation that all c^2 votes belong to the same voter
- A summation matrix is used to declare the winner

References I

David Chaum.

Secret-ballot receipts: True voter-verifiable elections.

Security & Privacy, IEEE, 2(1):38–47, 2004.

Michael R. Clarkson and Andrew C. Myers.

Coercion-resistant remote voting using decryption mixes.

In *In Frontiers in Electronic Elections (FEE 2005*, 2005.

David Chaum, Peter Y. A. Ryan, and Steve A. Schneider.

A practical voter-verifiable election scheme.

In *ESORICS*, pages 118–139, 2005.

James Heather.

Implementing stv securely in pret a voter.

In *Computer Security Foundations Symposium, 2007. CSF'07. 20th IEEE*, pages 157–169. IEEE, 2007.

Moni Naor and Adi Shamir.

Visual cryptography.

In *Advances in Cryptology□EUROCRYPT'94*, pages 1–12. Springer, 1995.

Peter YA Ryan, David Bismark, JA Heather, Steve A Schneider, and Zhe Xia.

The prêt à voter verifiable election system.

IEEE transactions on information forensics and security, 4(4):662–673, 2009.

Ronald L. Rivest.

On the notion of □software independence□in voting systems.

Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 366(1881):3759–3767, 2008.

References II

Peter Y. A. Ryan and Steve A. Schneider.

Prêt à voter with re-encryption mixes.

In *ESORICS*, pages 313–326, 2006.

Peter YA Ryan.

Prêt à voter with paillier encryption.

Mathematical and Computer Modelling, 48(9):1646–1662, 2008.

Zhe Xia, Chris Culnane, James Heather, Hugo Jonker, Peter YA Ryan, Steve Schneider, and Sriramkrishnan Srinivasan.

Versatile prêt à voter: Handling multiple election methods with a unified interface.

In *Progress in Cryptology-INDOCRYPT 2010*, pages 98–114. Springer, 2010.