Ballot permutations in Prêt à Voter

Vanessa Teague
University of Melbourne
Joint work with Peter Y A Ryan
Luxembourg University

Summary

- This talk is about how we should construct the candidate order in Prêt à Voter
  - There are lots of alternatives available
  - This is one more, arguably the best
  - It's only good for selecting one candidate
    - Not for STV, IRV, AV etc.

Outline

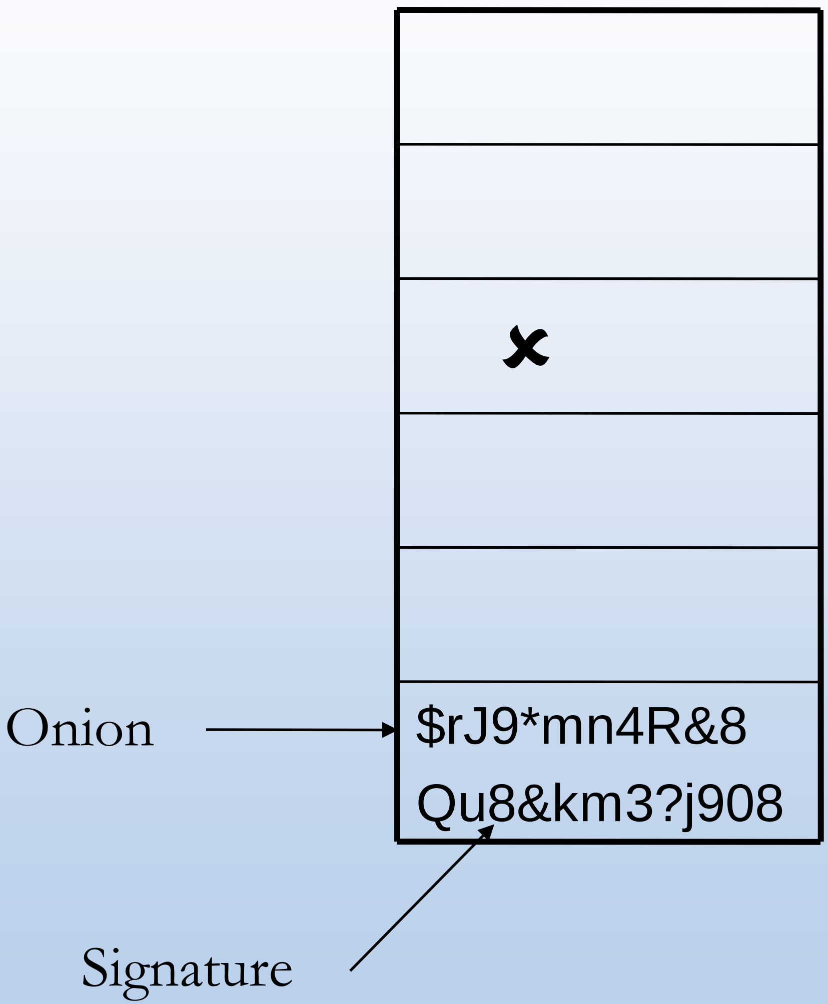
- Intro to Prêt à Voter
- Existing ways of generating the candidate ordering
- Some issues in some circumstances
- Our solution
  - For prime numbers of candidates
  - For composites

Prêt à Voter

- Uses pre-prepared ballot forms that encode the vote in familiar form.
- The candidate list is **randomised** for each ballot form.
- Information defining the candidate list is encrypted in an “onion” value printed on each ballot form.

Red	
Green	✗
Chequered	
Fuzzy	
Cross	
	$rJ9*mn4R&8

Voter's Ballot Receipt



- Various procedures to ensure the onion
  - Matches the candidate list
  - Doesn't leak the candidate list (except with the right key)
- Tallying on a bulletin board
  - With proof of correctness

Outline

- Intro to Prêt à Voter
- Existing ways of generating the candidate ordering
- Some issues in some circumstances
- Our solution
  - For prime numbers of candidates
  - For composites

Existing ways of randomising the candidate list

1. Print one ciphertext per candidate [PaV05, Scratch & Vote, Xia et al EVT08]
   - But might use too much space
2. Use cyclic shifts of a fixed order [Pav06]
   - But depends on voter vigilance to verify checkmarks aren't shifted
3. Use a single ciphertext to encode a random permutation [PaVwithPaillier08]
   - But decryption on the BB may violate privacy

Full permutations in one ciphertext

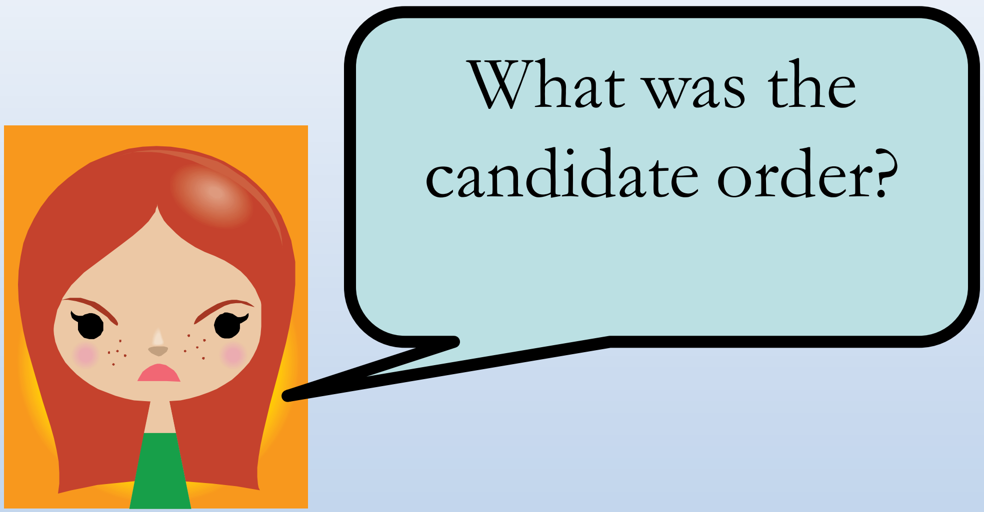
- Could we write a full permutation, but in one ciphertext?
  - Mix all the ({permutation}, {index}) pairs
  - Decrypt the permutation on the BB and derive the selected candidate name
  - Vulnerable to a pattern-recognition (a.k.a. “Italian”) attack when there are lots of candidates, even for first-past-the-post
  - Adding a cyclic shift, as in [PaVwithPaillier08], doesn’t fix it

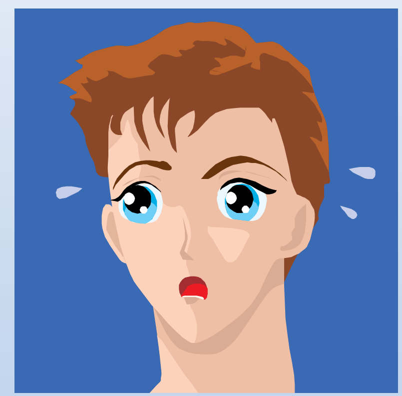
Outline

- Intro to Prêt à Voter
- Existing ways of generating the candidate ordering
- Some issues in some circumstances
- Our solution
  - For prime numbers of candidates
  - For composites

Full permutations in one ciphertext (con't)

– The coercer visits the voter after he votes but before tallying, and demands to know his ballot permutation





– The voter could lie, but...

Full permutations in one ciphertext (con't)

- When the permutations are decrypted on the BB, the coercer
  - Looks for the claimed ballot permutation
    - **If n! > #voters, there's only likely to be one vote consistent with the voter's story**
    - **Or 0 if he lied**
  - Sees which candidate was chosen
  - Rewards or punishes the voter
  - (If the voter somehow knows another tabulated permutation, he can resist coercion)

Cyclic shifts vs “defence in depth”

- Perfectly hiding, but reliant on some voter vigilance
- if an attacker can manipulate some checkmarks undetected, she can **systematically** skew the outcome.
  - e.g. if Green is always two steps after Red, attack a precinct where everyone votes Green and shift checkmarks 2 steps to benefit Red
  - Benaloh's hash chain of receipts would fix this
    - except the immediate input attack

Outline

- Intro to Prêt à Voter
- Existing ways of generating the candidate ordering
- Some issues in some circumstances
- Our solution
  - For prime numbers of candidates
  - For composites

Florentine squares

- Key property:
- For any two distinct candidates A and B and for any shift t, there exists exactly one row such that A and B are separated by t.
- So, assuming that the adversary doesn't know the row, shifting the X is equally likely to produce any other candidate.

Using Florentine squares

- Florentine squares are well known and easy to construct when n is prime
  - (n = #candidates)
- C = k.i mod n
  - C = candidate,
  - k = row,
  - i = column

0	1	2	3	4	5	6
0	2	4	6	1	3	5
0	3	6	2	5	1	4
0	4	1	5	2	6	3
0	5	3	1	6	4	2
0	6	5	4	3	2	1

Using Florentine squares

- We still need a cyclic shift s
- Now each ballot has two onions:
  - {k} $k \in [1,n-1]$, the row of the Florentine square
  - {s} $s \in Z_q^*$, a cyclic shift.
- The candidate order will be given by the k-th row shifted cyclically upwards by:

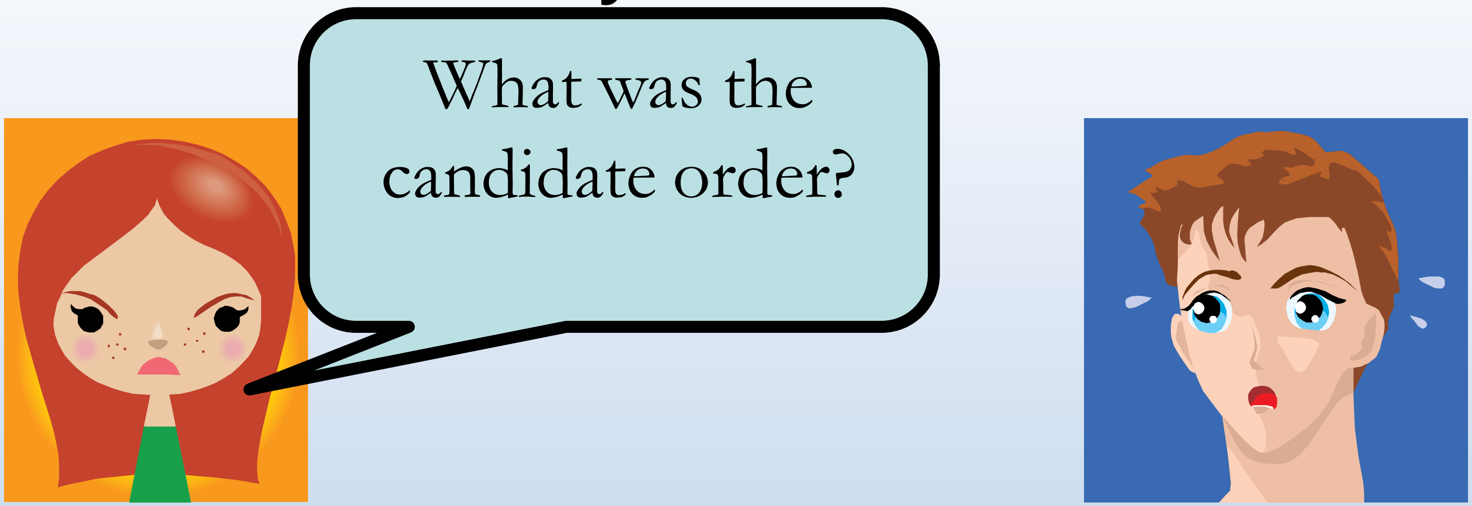
$$k^{-1} s \quad (\text{mod } n)$$

Extracting and tallying the vote

- Thus, for a ballot with k and s, for which the voter chooses index i, their candidate will be:
- $i \cdot k + s \pmod{n}$
- Thus we can transform the receipt
  - (i, {k}, {s})
  - Using the additive homomorphism $\oplus$
  - To $i\{k\} \oplus \{s\} = \{i \cdot k + s\}$
- Which can be put through mixes.

Receipt freeness

- The coercer can try the same attack



- But the voter just lies about the cyclic shift
  - Pretends that the true ballot permutation was whatever he really received, shifted to please the coercer

Non-prime numbers of candidates

- We could just pad it out with NULL candidates, or
- Construct the ballot permutation from F_p, where p is the largest prime less than n
  - Choose a random row of F_p
  - Insert p+1, p+2, ... in random places until enough candidates
  - Apply a cyclic shift

Non-prime numbers of candidates (con't)

- Now there are 2 + (n-p) ciphertexts on the ballot
  - (n is the number of candidates, p the nearest smaller prime)
- This retains the symmetry property
  - so shifting the checkmark produces no systematic shift from one candidate to another

Non-prime numbers of candidates
Privacy and tabulation

- The tabulation reveals some, but not much, info about the candidate selection
  - Whether the candidate came from the Florentine square part or not,
  - but equally likely to be any candidate
- A coercer may try the pattern-based attack
  – But again the voter just lies about the cyclic shift

Attack models

- This seems to counter the skewing attack, or at least ensure that the attacker can at best randomise votes.
- But no good if she tries to manipulate the k and s onions
- This seems best countered by applying signatures to these and perhaps pre-posting them to the WBB.
- Note: we can pre-audit such signatures, in contrast to the signatures on the receipts.

Further work

- Other voting schemes
  - AV, STV, etc