Estonian Voting Verification Mechanism Revisited Again

Ivo Kubjas[1], Tiit Pikma[1], Jan Willemson[23]

[1]Smartmatic-Cybernetica Centre of Excellence for Internet Voting
[2]Cybernetica
[3]Software Technology and Applications Competence Centre

October 26th, 2017

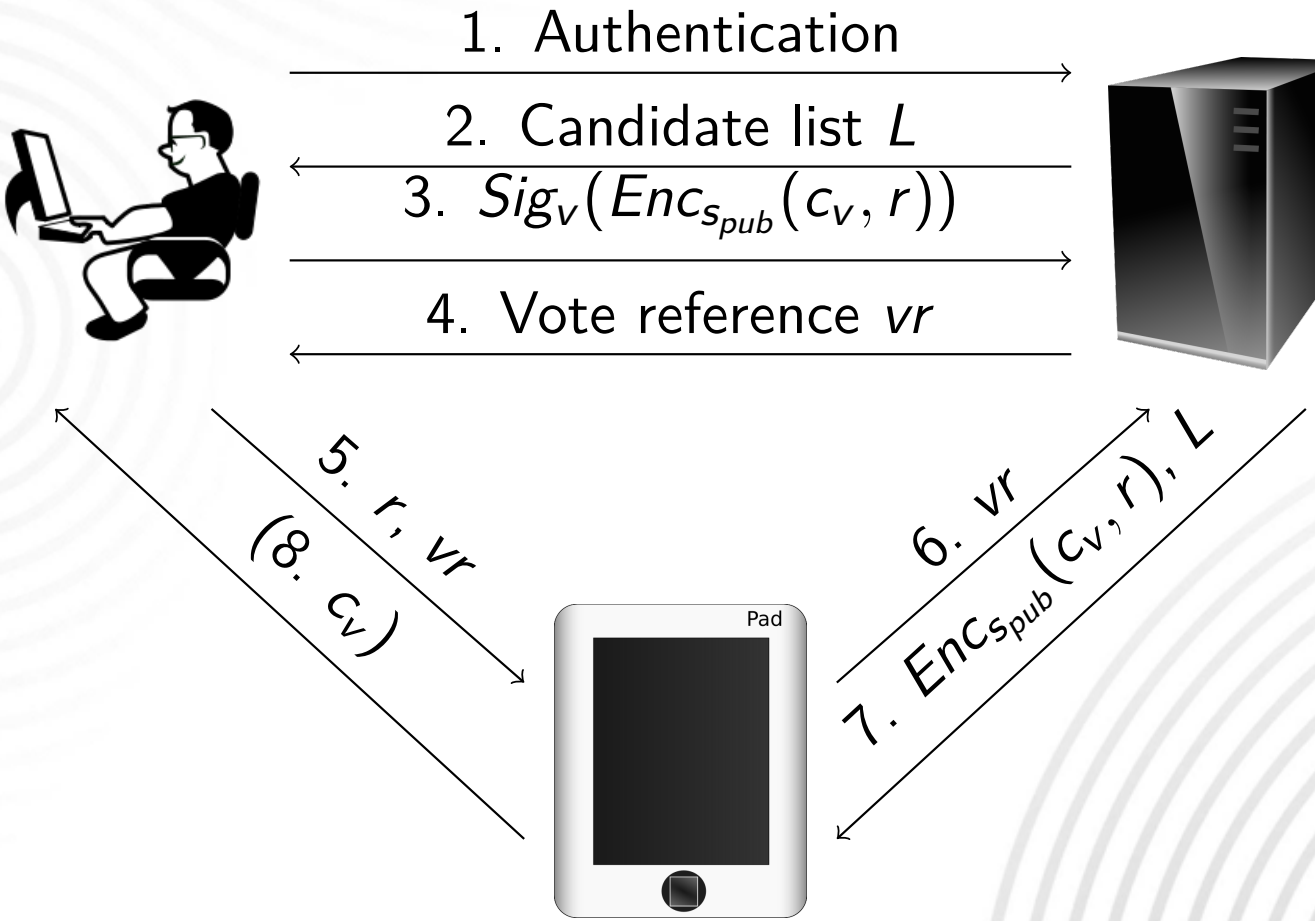
Estonian Internet voting scheme

1. Authentication →

← 2. Candidate list L

3. $Sig_v(Enc_{s_{pub}}(c_v, r))$ →

Estonian Internet voting scheme



SMARTMATIC CYBERNETICA

Criticism by Muș *et al.*

- In December 2016, Muș, Kiraz, Cenk and Sertkaya critisised weak privacy properties of the Estonian vote verification scheme
  - Verification device learns the voter preference
  - If it also learns the voter identity, it will have broken the vote privacy requirement
    - Up to 2015, the verification app actually did not learn the voter identity
    - But in 2017 it did
  - The authors proposed an updated scheme
  - In this paper we show that their scheme is broken in several ways and fix it a bit

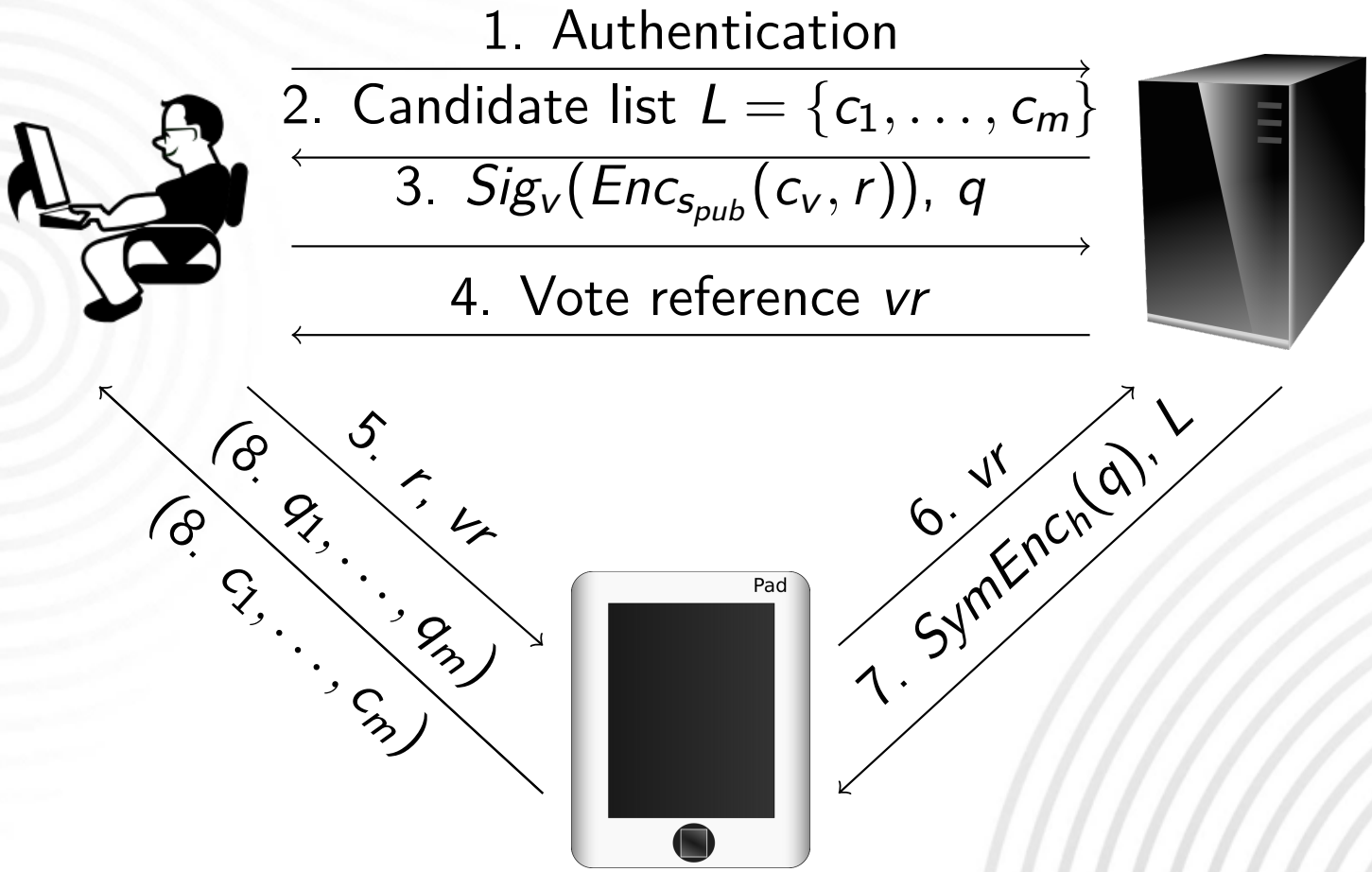
Proposed update by Muș *et al.*

- Basic idea: let the voter pick a random verification code q and send it to server together with encrypted ballot
- Compute hash of the vote cryptogram $h = H(Enc_{s_{pub}}(c_v, r))$ and use it to encrypt q as $SymEnc_h(q)$
- The verification app goes through the candidate list $L = \{c_1, c_2, \ldots, c_m\}$ and computes $h_i = H(Enc_{s_{pub}}(c_i, r))$ for every candidate c_i
- Next, the app computes verification code candidates as

$$q_i = SymDec_{h_i}(SymEnc_h(q)) \quad (i = 1, 2, \ldots, m)$$

- Last, the app displays pairs (c_i, q_i)
- The voter should see her own code q next to her choice c_v; the other codes look random

Proposed scheme



SMARTMATIC CYBERNETICA

Usability issues

- Asking for a long random q is unrealistic
  - Humans are poor RNGs
  - User interface for randomness input is problematic (the voter must be able to recognise it later!)
- Hence, the proposal includes requesting only 32-bit q_{right}
  - In practice, this means asking for 4 symbols
- The voting client will generate an additional random value q_{left} and set $q = q_{\text{left}} || q_{\text{right}}$
- Only 32 rightmost bits of q_i will be displayed by the verification app

1st attack: privacy

- Key observation: not every byte has a common symbol representation
  - In fact, only about 75 out of 256 do
- Rightmost 4 bytes of wrong q_i-s are random
- The probability that at least one of these bytes falls outside of the 75 symbol set is

$$1 - \left(1 - \frac{256 - 75}{256}\right)^4 \approx 0.993$$

- Hence, wrong q_i-s are easily recognisable with high probability
- This defeats the purpose of protecting the voter preference from a malicious verification app

2nd attack: vote manipulation

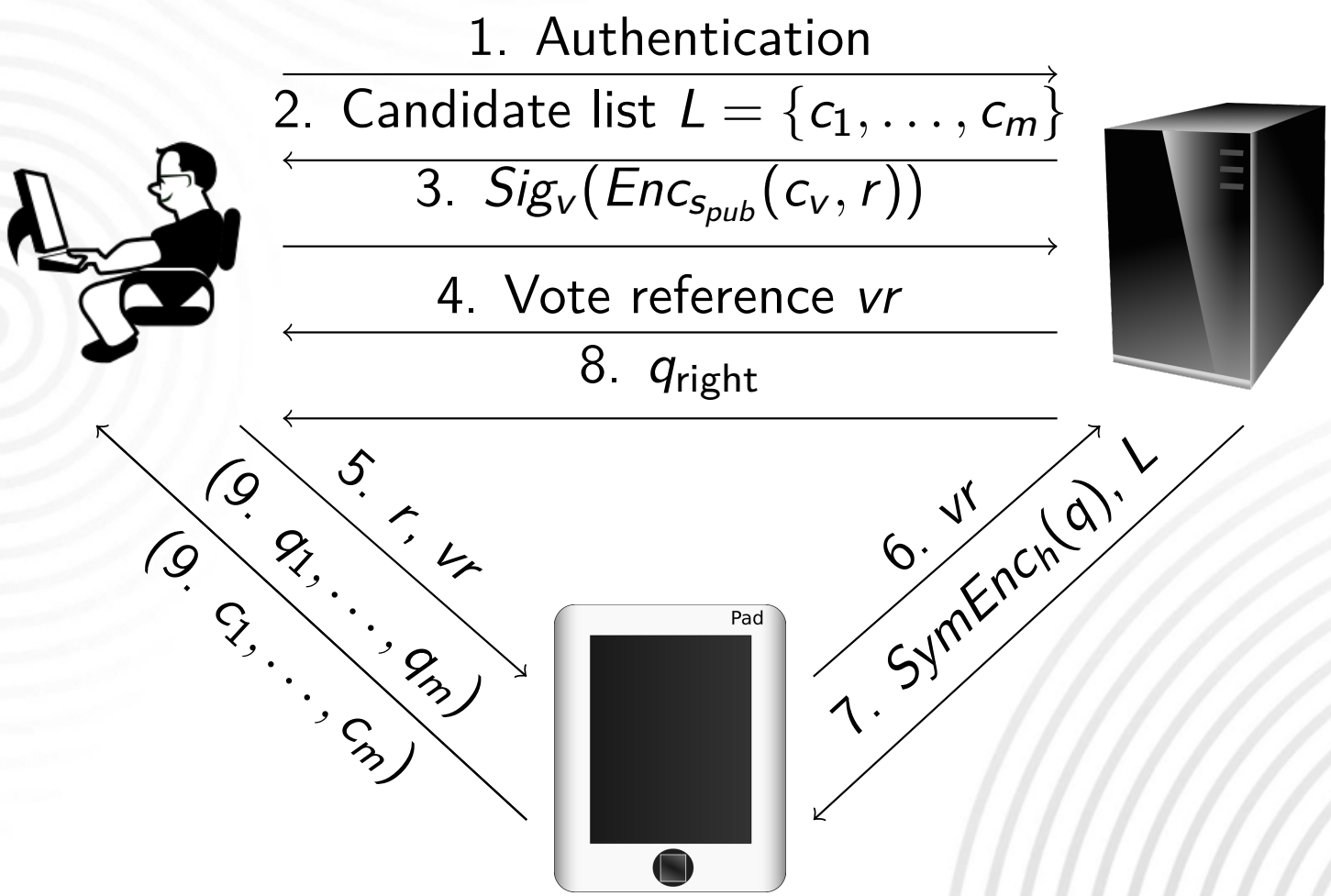
- Key observation: a malicious voting app has a lot of freedom choosing r and q_{left} non-randomly
- Attack scenario: the attacker wants to give votes to his preferred candidate c_j
  - Having acces to c_v and q_{right}, he can fix q and choose r so that the encrypted vote is c_j, but the rightmost 4 bytes of q_v are exactly q_{right}
  - The choice can be precomputed to a large table
  - The size of the table is in the order of magnitude of gigabytes
  - Allowing $2^{34}m$ computations (where m is the number of candidates) allows us to fill 98% of the table
  - For details, see the paper

q_{right}	choice c_i	$r_{i,q_{\text{right}}}$
0	c_1	$r_{1,0}$
⋮	⋮	⋮
$2^{32}-1$	c_1	$r_{1,2^{32}-1}$
0	c_2	$r_{2,0}$
⋮	⋮	⋮
$2^{32}-1$	c_2	$r_{2,2^{32}-1}$
⋮	⋮	⋮
0	c_m	$r_{m,0}$
⋮	⋮	⋮
$2^{32}-1$	c_m	$r_{m,2^{32}-1}$

Improving the proposal

- Key observation: both of the described risks can be mitigated if the server selects q and makes q_{right} available to the voting app only *after* it has committed to the vote cryptogram
  - Privacy violation is possible if q_{right} is available to the verification app, but this assumes malicious cooperation of the verification app and the server
  - Vote manipulation would require malicious cooperation of the voting app and the server
- Usability issues still remain
  - How to present 32 random bits to the voter?
  - Verification app can make educated guesses based on the point where the user stops scrolling through the pairs (c_i, q_i)

Improved protocol



SMARTMATIC CYBERNETICA

Conclusions

- Protecting vote privacy from the verification app is hard
- Usability issues seem to have direct implications on privacy
- Secure app development and publishing may resolve some issues