Kleptographic Weaknesses in Benaloh-Tuinstra Protocol

Piotr Borzęcki[1], Jędrzej Kabarowski, Przemysław Kubiak, Mirosław Kutyłowski, and Filip Zagórski

[1] Faculty of Mathematics and Computer Science, Wrocław University,

[2] Institute of Mathematics and Computer Science, Wrocław University of Technology, Wrocław, Poland

ICSNC'2006

What is wrong with traditional voting?

- paper ballots can be manipulated
- vote counting is unreliable
- personal appearance at voting booth - inconvenient and costly

Mail-in procedures

advantages :

- convenient for a voter

disadvantages :

- vote selling cannot be prevented
  case: local elections 2004 in UK
- insecure delivery – ballots can be removed from Post Office
  remove the ballots at random from districts where the competition has better chances, case US?

Demands on e-voting systems

- it should be convenient for a voter
- no special hardware required, standard PC's should suffice
- auditable

Motivations

anonymity voters preferences must remain hidden

your boss has friends in the election authority, they may say him how you have voted

case Brasilia and auditable paper traces

no vote selling a voter cannot prove how he votes

case Birmingham, selling votes for 1 pound in local elections

Demands on e-voting schemes

correctness the votes are counted honestly

it does not matter who casts the votes, it matters who counts them

verifiability a voter can check that her vote was counted

why to vote since my vote will be removed anyway, auditable paper traces

Verification approaches

Verifiability:

- global : correctness of the procedure as a whole can be checked
- local : one can check if his vote has been taken into account

Verification degrees

- open design
- Tiger teams
- limited verification (Germany!)

System components

Typical parts of the system are:

- voting machine or a private machine of the voter
- bulletin board(s)
- a network of mix servers:
  - encoded votes coming into the network
  - at each layer of the network: recoded and permuted

Main question:

Can e-voting protocols be designed independently from implementation dangers?

Can layered design approach be adapted?

Main question:

Can e-voting protocols be designed independently from implementation dangers?

Can layered design approach be adapted?

Main answer:

No!

countermeasures against implementation attacks must be found already during scheme design phase

Necessity of randomness in e-voting

- each tallying authority uses private keys to perform re-coding
- if the whole process deterministic:
  - perform trial encryptions with the public keys
  - compare with the partial results
  - voter's choice revealed immediately

Necessity of randomness in e-voting

- each tallying authority uses private keys to perform re-coding
- if the whole process deterministic:
  - perform trial encryptions with the public keys
  - compare with the partial results
  - voter's choice revealed immediately
- therefore voters' choices must be masked by (pseudo)random values
  an often situation in cryptographic protocols

Dangers of randomness

It is known that freedom of parameters valuation makes room for a *subliminal channel*, through which may leak:

- voters' choices,
- signing keys of voting machines,
- ...

Kleptography I

- discovered by Yung and Young ten years ago,
- implementation of “Big Brother” with only one TV receiver, while “Big Brother” remains perfectly hidden
  - the channel is protected (encrypted) by a public key of a malicious Mallet,
  - reading data from kleptographic channel with a secret key only,

Kleptography II

- input-output testing cannot detect klepto-code,
- reverse engineering of a device/software “compromises” only the public key, the private key is not there!
- how many tamper resistant cards/software copies can you check?

Kleptography III

Infection ways:

1. malware that alters a voting programs
2. malware that influences (pseudo) random generators
3. malware that attacks an operating system
   case: installing unsigned kernel drivers to Vista and virtual machine mode

A perfect technology for corrupting elections.

Encryption

- $n = pq$ like for RSA, p and q large primes
- r is a large prime factor of $p-1$, but not of $q-1$
- $y \in \mathbb{Z}_n^* (= \{1, \ldots n-1\})$ chosen so that $r | \mathrm{rank}(y)$, $\mathrm{rank}(y) = \min\{i : y^i \bmod n = 1\}$

Encryption

- $n = pq$ like for RSA, p and q large primes
- r is a large prime factor of $p-1$, but not of $q-1$
- $y \in \mathbb{Z}_n^*(= \{1, \ldots n-1\})$ chosen so that $r | \mathrm{rank}(y)$, $\mathrm{rank}(y) = \min\{i : y^i \bmod n = 1\}$
- plaintext $x \in \{0, 1, \ldots, r-1\}$,

Encryption

- $n = pq$ like for RSA, p and q large primes
- r is a large prime factor of $p-1$, but not of $q-1$
- $y \in \mathbb{Z}_n^*(= \{1, \ldots n-1\})$ chosen so that $r | \mathrm{rank}(y)$, $\mathrm{rank}(y) = \min\{i : y^i \bmod n = 1\}$
- plaintext $x \in \{0, 1, \ldots, r-1\}$,
- set of ciphertexts of x is

$$E(x) = \{\theta^r y^x \bmod n : \theta \in \mathbb{Z}_n^*\}$$

$$E(x) = \{\theta^r y^x \bmod n : \theta \in \mathbb{Z}_n^*\}$$

- θ^r blinds y^x in a clever way, nevertheless it does not change the component of rank r of y^x mod n

$$E(x) = \{\theta^r y^x \bmod n : \theta \in \mathbb{Z}_n^*\}$$

- θ^r blinds y^x in a clever way, nevertheless it does not change the component of rank r of $y^x \bmod n$
- "homomorphic" operator $\otimes$:

$$z_1 \otimes z_2 = (z_1 z_2) \bmod n$$

If $z_1 \in E(x_1)$, $z_2 \in E(x_2)$, then

$$z_1 \otimes z_2 \in E((x_1 + x_2) \bmod r)$$

Single Authority Election

elections setting:

- two candidates (Alice and Bob) only,
- two options: vote for Alice or vote for Bob,
- it suffices to count the number of votes cast for Alice

Single Authority Election - voting procedure

1. each voter sends a ciphertext $c \in E(x)$ of her/his choice x

Single Authority Election - voting procedure

1. each voter sends a ciphertext $c \in E(x)$ of her/his choice x
2. all votes composed with $\otimes$, result:

$$\theta^r y^x = (\theta_1 \theta_2 \ldots \theta_N)^r y^{x_1 + x_2 + \ldots + x_N} \bmod n$$

where θ_i, x_i chosen by the ith voter

Single Authority Election - voting procedure

1. each voter sends a ciphertext $c \in E(x)$ of her/his choice x
2. all votes composed with $\otimes$, result:

$$\theta^r y^x = (\theta_1 \theta_2 \ldots \theta_N)^r y^{x_1 + x_2 + \ldots + x_N} \bmod n$$

   where θ_i, x_i chosen by the ith voter
3. the authority computes:

$$(\theta^r y^x)^{(p-1)(q-1)/r}$$

   and compares the result with

$$(y^i)^{(p-1)(q-1)/r}$$

   for each i.
   recall: $(\theta^r)^{(p-1)(q-1)/r} = \theta^{(p-1)(q-1)/} = 1 \bmod n$

Ballot preparation

- ballots prepared in advance by some authority, each ballot consists of a number of pairs of ciphertexts

$$\alpha_i, \beta_i$$

  one of them from $E(0)$, one from $E(1)$.
- cut and choose verification procedure: proved that each pair contains a ciphertext of 0 and a ciphertext of 1

Attack

- observe that:

$$\alpha_0 \cdot \beta_0 = (\theta_0 \cdot \theta_1)^r y \bmod n$$

- infected software takes:

$$\theta_0 = g^{k_1} \bmod n, \quad \theta_1 = g^{k_2} \bmod n$$

then

$$\alpha_0 \cdot \beta_0 / y = (g^{k_1+k_2})^r \bmod n$$

- malware uses public key $Y = g^s \bmod$:
  random choices derived by a pseudorandom generator with seed

$$\mathrm{HASH}((Y^{k_1+k_2})^r)$$

Retrieving voter's choices

1. Mallet takes private key s and computes

$$u := (\alpha_0 \cdot \beta_0 / y)^s \bmod n$$

Retrieving voter's choices

1. Mallet takes private key s and computes

$$u := (\alpha_0 \cdot \beta_0 / y)^s \bmod n$$

2. Mallet initializes the pseudorandom generator with $H(u)$,

Retrieving voter's choices

1. Mallet takes private key s and computes

$$u := (\alpha_0 \cdot \beta_0 / y)^s \bmod n$$

2. Mallet initializes the pseudorandom generator with $H(u)$,
3. observe that that

$$u = (\alpha_0 \cdot \beta_0 / y)^s = ((g^{k_1+k_2})^r)^s = ((g^s)^{k_1+k_2})^r = (Y^{k_1+k_2})^r \bmod n$$

so Mallet will know the choices of the malware (no communication necessary)

Retrieving voter's choices

1. Mallet takes private key s and computes

$$u := (\alpha_0 \cdot \beta_0 / y)^s \bmod n$$

2. Mallet initializes the pseudorandom generator with $H(u)$,
3. observe that that

$$u = (\alpha_0 \cdot \beta_0 / y)^s = ((g^{k_1+k_2})^r)^s = ((g^s)^{k_1+k_2})^r = (Y^{k_1+k_2})^r \bmod n$$

   so Mallet will know the choices of the malware (no communication necessary)
4. Mallet knows θ from the ballot component

$$\theta^r y^j \bmod$$

   and can find j, since $j \in \{0, 1\}$

Efficient implementation

- the method of Möller
- a twisted pair of elliptic curves used

Algebraic weakness

- probability distribution of (Y^k) is not uniform within $\mathbb{Z}_n^*$,
- reason – algebraic structure of $\mathbb{Z}_n^*$
  it is a product of groups, some of them small - a component of g in one of these groups can be 1,
  then all $\theta_0 = g^k$ have this component equal to 1
- observable through restarting the device
- Mallet does not know factorization of n, so cannot choose g appropriately

Hiding the attack

- instead of a single g use a list $g_1, \ldots g_L$
- after initialization choose $g_{i_1} \cdot \ldots \cdot g_{i_M}$ at random, and sets:

$$g_{i_1} \cdot \ldots \cdot g_{i_M} \bmod n$$

g is used as before

this hiding technique can be used for other klepto-attacks

Statistical effectiveness

a practical countermeasure: statistical irregularities still exist, but finding them requires enormous amount of data

The countermeasure: deterministic random numbers

- avoid unnecessary randomness
- produce random values from signatures (in Chaum's manner):

$$r = \mathcal{R}(\mathrm{sig}(h(q))),$$

where:

- $\mathcal{R}$ is a strong pseudorandom number generator,
- sig is a deterministic signature scheme,
- h is a cryptographically strong hash function,
- q is a deterministic parameter (sequential number ...)

Problem

- this requires a complete re-design of the election protocol

Conclusions

A critical requirement for e-voting systems:

... for the product offered there must be an evidence that the system proposed is immune against malware ...

at the moment elections via Internet seem to be too risky