Kleptographic Attacks on a Cascade of Mix Servers

Przemysław Kubiak, Mirosław Kutyłowski, Filip Zagórski

Institute of Mathematics and Computer Science
Wrocaw University of Technology

ASIACCS, Singapore 2007

Anonymity

In some applications anonymous communication is important:

Anonymity

In some applications anonymous communication is important:

- electronic auctions: creating a profile of a bidder based on previous auctions could help to predict the bidders decisions, thus good protocols require non-profilability,

Anonymity

In some applications anonymous communication is important:

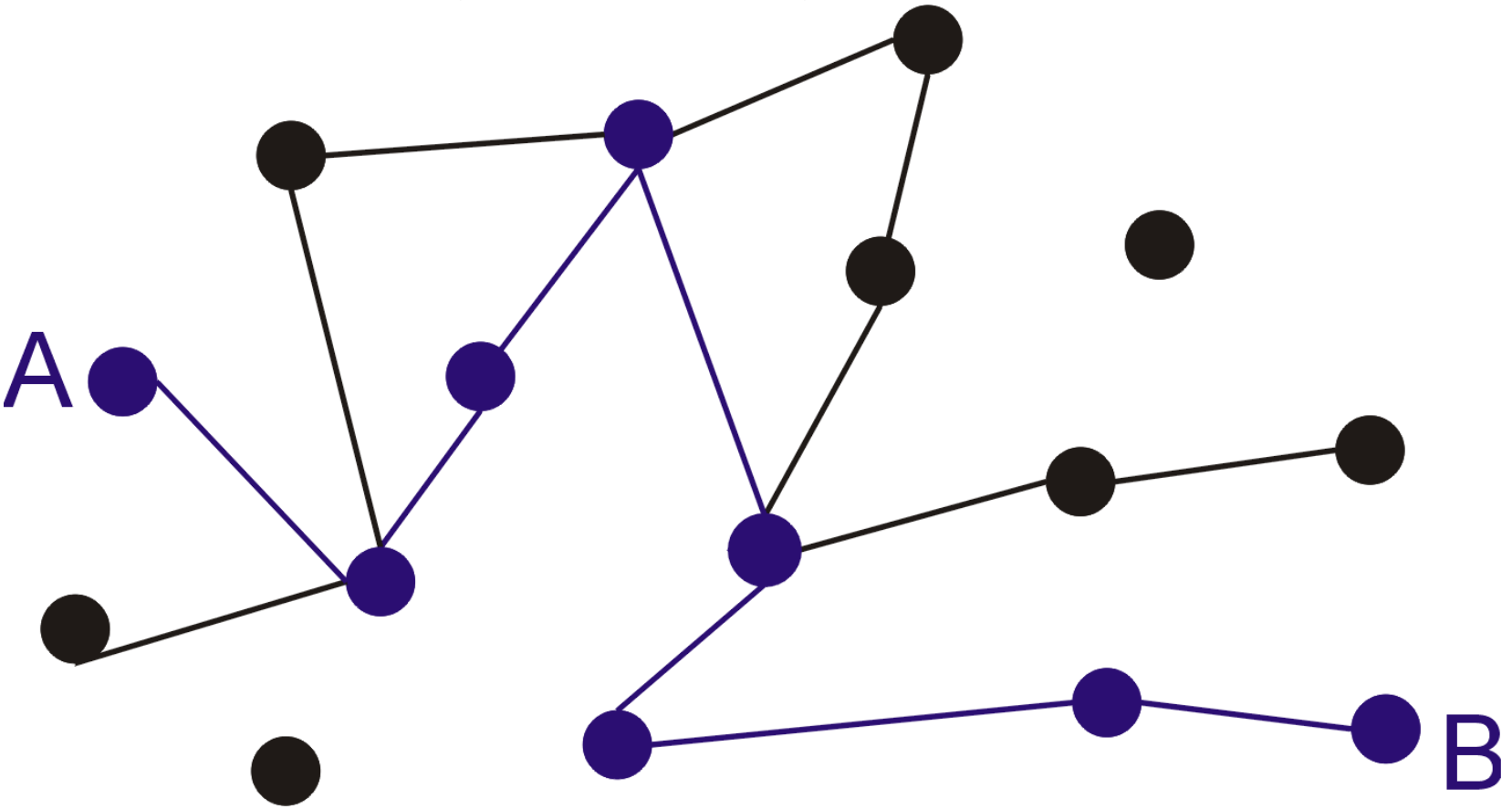
- electronic auctions: creating a profile of a bidder based on previous auctions could help to predict the bidders decisions, thus good protocols require non-profilability,
- electronic voting: votes must be cast anonymously,

Anonymity

In some applications anonymous communication is important:

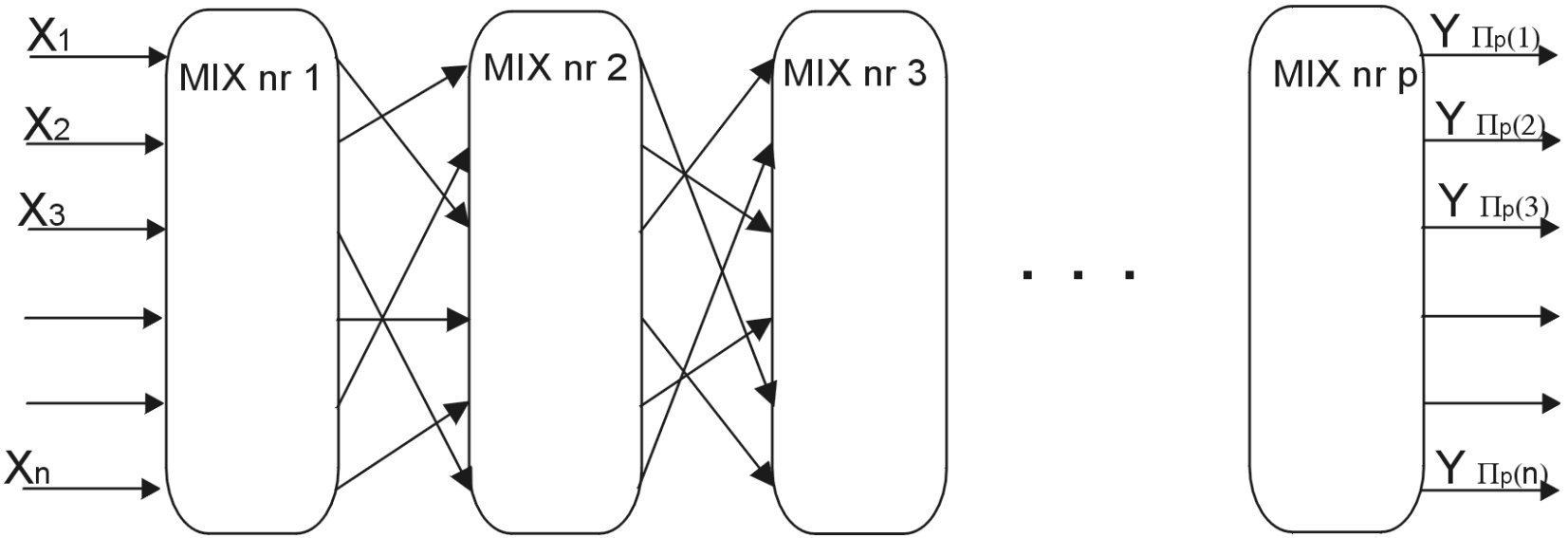
- electronic auctions: creating a profile of a bidder based on previous auctions could help to predict the bidders decisions, thus good protocols require non-profilability,
- electronic voting: votes must be cast anonymously,
- other, like counteracting industrial intelligence.

Roughly speaking, there are two main types of anonymizing networks:
1) the onion routing networks (e.g. TOR)



...Anonymizing networks

2) and cascades of mix servers (like AN.ON or cascades being a part of e-voting protocols)



Cascades of Mix Servers

- In the paper we present kleptographic attacks on cascades of mix servers.
- Most of the attacks might be applied to any probabilistic cascade, but we focus on a cascade used as a building block of the Prêt à Voter (PaV) e-voting protocol:

  [PaV] P. Y. A. Ryan and S. A. Schneider. *Prêt à Voter with re-encryption mixes*. In: ESORICS, LNCS 4189, pages 313-326. Springer, 2006.
- The attacks assume that to reduce costs and simplify service or buying procedures the mix servers are taken from the same manufacturer, or the cascade is installed or supervised by the same entity.

Kleptography

- Designed by Moti Yung and Adam Young about ten years ago.

Kleptography

- Designed by Moti Yung and Adam Young about ten years ago.
- exploits randomness present in the protocol and builds a subliminal channel that is protected (encrypted) by a public key of an addressee of the subliminal message,

Kleptography

- Designed by Moti Yung and Adam Young about ten years ago.
- exploits randomness present in the protocol and builds a subliminal channel that is protected (encrypted) by a public key of an addressee of the subliminal message,
- reading data from kleptographic channel with a secret key only,

Kleptography

- Designed by Moti Yung and Adam Young about ten years ago.
- exploits randomness present in the protocol and builds a subliminal channel that is protected (encrypted) by a public key of an addressee of the subliminal message,
- reading data from kleptographic channel with a secret key only,
- non-invasive testing cannot detect klepto-code (kleptography works on code-level, the protocol is not changed!),

Kleptography

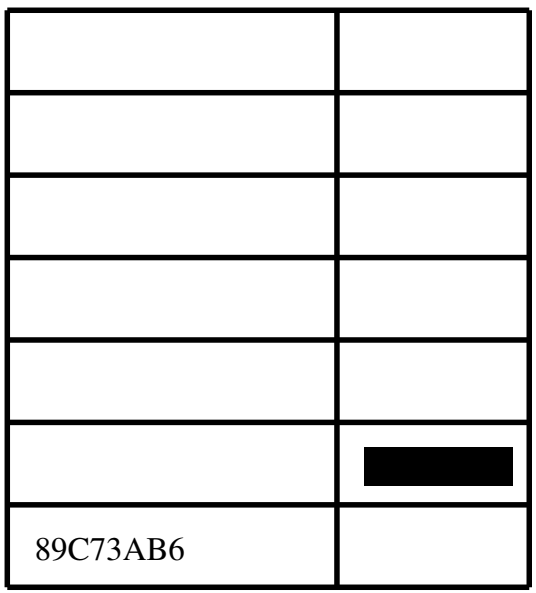
- Designed by Moti Yung and Adam Young about ten years ago.
- exploits randomness present in the protocol and builds a subliminal channel that is protected (encrypted) by a public key of an addressee of the subliminal message,
- reading data from kleptographic channel with a secret key only,
- non-invasive testing cannot detect klepto-code (kleptography works on code-level, the protocol is not changed!),
- reverse engineering of a device/software “compromises” only the public key, the private key is not there!

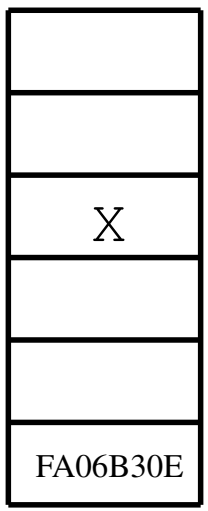
The Parties in the PaV

There are the following parties in the system

- clerks – authorities responsible for onions generation, each clerk leads a single re-encryption mix server; the paper [PaV] implicitly does not distinguish between the clerks and the servers; we shall make a distinction between these two notions, thus we are able to take the role of system's manufacturer into account;
- registrars – authorities responsible for delivering lists of candidates,
- tellers – authorities responsible for decoding ballots.

1. A set of λ clerks prepares a batch of pairs of onions – both onions in each pair j are ciphertexts of the same plaintext $S_{\lambda,j}$, the exemplary pair is (89C73AB6,FA06B30E).
2. The last clerk prints proto-ballots, with right-hand side onions concealed under scratch strips.
3. The registrars decrypt the left-hand side onion, on the basis of $S_{\lambda,j}$ the order of the candidates is determined, the names are printed and the bottom row is removed.
4. A voter marks an X against her candidate and the left-hand strip is removed. Later the scratch strip is removed as well.





The Cascade of Clerks...

The mix server of the first clerk outputs a batch of N pairs of the form:

$$\left((g^{K_{1,j,R}}, S_{1,j} \cdot \beta_R^{K_{1,j,R}}) \, , \, (g^{K_{1,j,T}}, S_{1,j} \cdot \beta_T^{K_{1,j,T}}) \right) . \qquad (1)$$

where

- $j = 1, 2, \ldots, N$,
- $S_{1,j} = g^{s_{1,j}}$ for $s_{1,j}$ being integers of small absolute value,
- $\beta_R = g^{x_R}$ is the public key of the registrars,
- β_T is the public key of the tellers.

...The Cascade of Clerks

For every $i > 1$, the ith server takes its input batch and for each pair re-computes the ciphertexts:

$$g^{K_{i,j,\alpha}} := g^{K_{i-1,j,\alpha}} \cdot g^{k_{i,j,\alpha}}, \quad (2)$$

$$\beta_\alpha^{K_{i,j,\alpha}} := \beta_\alpha^{K_{i-1,j,\alpha}} \cdot \beta_\alpha^{k_{i,j,\alpha}}, \quad (3)$$

$$S_{i,j} := S_{i-1,j} \cdot g^{s_{i,j}} \quad (4)$$

for $\alpha \in \{R, T\}$, random $k_{i,j,\alpha}$, and small random $s_{i,j}$.
Hence

- $S_{i,j} = g^{\sum_{t=1}^{i} s_{t,j}}$,
- $K_{i,j,\alpha} = \sum_{t=1}^{i} k_{t,j,\alpha}$.

After performing steps (2) – (4), the last clerk λ produces proto-ballots.

Assumptions

- Output batches will be available on the Bulletin Board ($\mathcal{BB}$) after ballot cards generation. This reduces amount of information available for malicious mix servers (only to its input batch).
- For the sake of kleptographic cooperation each server holds an ElGamal "public" key $Y_i = G^{x_i}$ (public in the sense that each server i knows the keys Y_{i-1}, Y_{i+1} of its neighbors) and the corresponding private key x_i.

At first we analyze a version of the PaV protocol with single onions

$$O_{i,j} = (g^{K_{i,j}}, S_{i,j} \cdot \beta_T^{K_{i,j}}) \tag{5}$$

The Encoding Technique...

In the attack each single onion sent by server i to server $i+1$ carries a piece of information on some element $a := G^k$, where number k is chosen at random by server i
(G^k is never transmitted as such, for instance as one of two components of ElGamal ciphertexts).
If it succeeds, then both servers can compute

$$G^{k \cdot x_{i+1}} = (G^k)^{x_{i+1}} = Y_{i+1}^k.$$

Define

$$n := \lfloor \log_2 a \rfloor + 1$$

and let

$$(a_{n-1} a_{n-2} \ldots a_0)_2$$

be consecutive bits of a.

... The Encoding Technique

Define

$$\begin{cases} f_{address}(x) &= H_1(x) \bmod n, \\ f_{value}(x) &= H_2(x) \bmod 2 \end{cases} \tag{6}$$

The encoding technique was used in [YoungYung04] and is called the *Coupon Collector's Problem*. For each j the server i changes the value

$$\gamma_j = g^{K_{i-1,j}+k_{i,j}},$$

i.e. the first component of ciphertext (5), by incrementing $k_{i,j}$ in (2) until the equality

$$(f_{address}(\gamma_j), f_{value}(\gamma_j)) = (\ell, a_\ell) \tag{7}$$

holds for some $\ell \in \{n-1, n-2, \ldots, 0\}$. We call γ_j *proper*, if it satisfies property (7).

1. Server i chooses $k < \mathrm{ord}G$ uniformly at random, and computes $a := G^k$ and $z_{i,k} := Y_{i+1}^k$.
2. For $j = 1, \ldots, N$, server i computes

$$k'_{i,j} := \mathcal{R}(H^{(j)}(z_{i,k})) \bmod \mathrm{ord}g, \tag{8}$$

where $H^{(j)}$ stands for the j-fold composition of $H \neq \mathcal{R}$.
3. Server i decodes an element $a' = G^{k'}$ from the input batch obtained from server $i-1$; procedure described above is used. Then it computes $z_{i-1,k'} = (a')^{x_i}$.
4. As we will see, on the basis on $z_{i-1,k'}$ server i can reconstruct all $K_{i-1,j}$ for $j = 1, \ldots, N$.
5. Server i computes

$$k_{i,j} := k'_{i,j} - K_{i-1,j} \bmod \mathrm{ord}g\ , \tag{9}$$

and performs re-encryptions. Observe that according to definition (9) of $k_{i,j}$, we have $K_{i,j} = k'_{i,j} \bmod \mathrm{ord}g$. Then server i increments $k_{i,j}$ (i.e. it increments $K_{i,j}$) to get proper γ_j. When all onions are processed, the value of a is described by the batch and the server $i+1$ will be able to reconstruct a and calculate the number $z_{i,k}$ needed in (8).

Consequences for PaV...

The attack on protocol version with the pairs instead of single onions does not differ at all. Now we have $2N$ onions, and initial exponents $K_{1,j,R}$, $K_{1,j,T}$ are independently generated for each onion in the pair, so the kleptographic channel is more capacious.

Note that the attack relays on the first coordinates of ElGamal ciphertexts only!

...Consequences for PaV

- If all the servers are provided by the same producer, the server λ may find out all the exponents $K_{\lambda-1,j}$. Hence it may retrieve $S_{\lambda-1,j}$.
- Elements $S_{\lambda,j}$ determine permutations of the list of candidates. Let $y_M = g^{x_M}$ be Mallet's public key. Then the mix λ chooses $k_{\lambda,j}$ so that $y_M^{K_{\lambda,j}}$ reduced mod v points out the row on the voting card, which contains the name of a candidate supported by Mallet.
- Then Mallet can check whether a vote has been cast for this candidate: he calculates

$$[(g^{K_{\lambda,j}})^{x_M}] \bmod v,$$

and compares the result with the index of row containing the sign "X" written by a voter.

Further Details Included in the Paper

- Improvements in efficiency of the kleptographic channel, e.g. an *Error Correcting Code Technique* (which is our modification of the Coupon Collector's Problem technique).
- Abandoning the assumption that mix servers know the puplic keys of their neighbours.
- Attacks on a general purpose ElGamal cascade.

Incorporate Verification of Randomness Into the Main Protocol

- diversification of software/mix servers sources is important,
- avoid unnecessary randomness
  (e.g. an output batch always put in lexicographic order),
- produce (pseudo)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 for example the value of the input onion.

Thank you for your attention!