Secure Initialization in Single-hop Radio Networks

Mirosław Kutyłowski   Wojciech Rutkowski

Wrocław University of Technology

partially supported by Polish Committee for Scientific Research

Overview

- Initialization problem
- Model Details
- Performance goals
- Adversary model
- New Algorithm
- Tricks
- Algorithm overview
- Future work

Initialization problem

Given a Single Hop Radio network, the stations have no ID's.

Goal:

- each station gets a number in the range 1..n
- each number is used exactly once

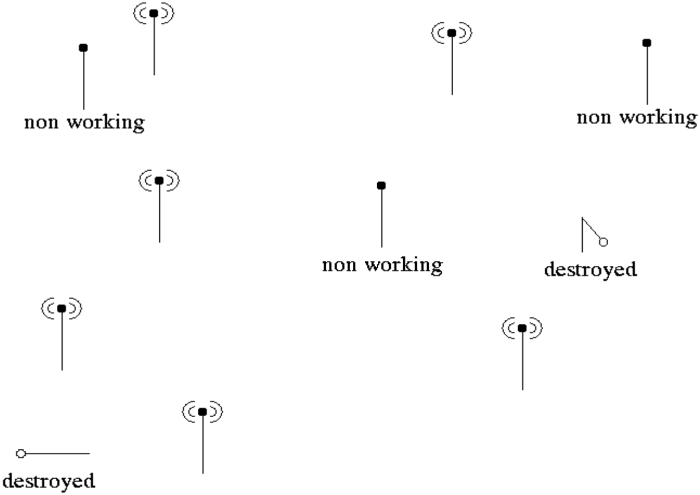
Initialization problem

Given a Single Hop Radio network, the stations have no ID's.
Goal:

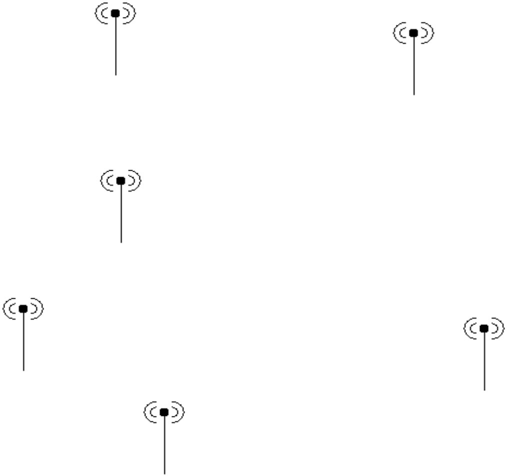
- each station gets a number in the range 1..n
- each number is used exactly once

Optimize for time and energy costs for each station.

Single-hop radio network



Single-hop radio network



Model Details

Single Hop Radio Network

(*RN*, *Ad Hoc network*)

- $O(N)$ processing units called *stations*
  N might be unknown – *size approximation problem*

Model Details

Single Hop Radio Network

(*RN*, *Ad Hoc network*)

- $O(N)$ processing units called *stations*
  N might be unknown – *size approximation problem*
- the stations **are not** numbered by consecutive numbers
  initialization problem

Model Details

Single Hop Radio Network

(*RN*, *Ad Hoc network*)

- $O(N)$ processing units called *stations*
  N might be unknown – *size approximation problem*
- the stations **are not** numbered by consecutive numbers
  initialization problem
- a single communication channel,
- messages sent simultaneously collide producing a random noise

Model Details

Single Hop Radio Network

(*RN*, *Ad Hoc network*)

- $O(N)$ processing units called *stations*
  N might be unknown – *size approximation problem*
- the stations **are not** numbered by consecutive numbers
  initialization problem
- a single communication channel,
- messages sent simultaneously collide producing a random noise
- stations cannot detect collisions
  no collision detection model – no-CD,

Model Details

Single Hop Radio Network

(*RN*, *Ad Hoc network*)

- $O(N)$ processing units called *stations*
  N might be unknown – *size approximation problem*
- the stations **are not** numbered by consecutive numbers
  initialization problem
- a single communication channel,
- messages sent simultaneously collide producing a random noise
- stations cannot detect collisions
  no collision detection model – no-CD,
- discrete, synchronous *time slots*,

Performance Measures

time - the number of time slots

Performance Measures

time - the number of time slots

energy cost - the maximal number k such that some station transmits **or** listens k times during algorithm execution

Performance Measures

time - the number of time slots

energy cost - the maximal number k such that some station transmits **or** listens k times during algorithm execution

- communication consumes almost all energy used
- energy required for transmitting and listening of the same magnitude (processor and sensors usage - negligible)
- battery exhaustion issue
- **extremely important for practical reasons!**

Adversary model

- random transmission errors, or burst errors, or even a malicious adversary knowing the algorithm

Adversary model

- random transmission errors, or burst errors, or even a malicious adversary knowing the algorithm
- legitimate stations have a secret that is not known by the adversary
  $\Rightarrow$ keyed MAC can be used to prevent faking messages by an adversary

Adversary model

- random transmission errors, or burst errors, or even a malicious adversary knowing the algorithm
- legitimate stations have a secret that is not known by the adversary
  $\Rightarrow$ keyed MAC can be used to prevent faking messages by an adversary
- the adversary attempts to cause collisions so that the algorithm fails

Adversary model

- random transmission errors, or burst errors, or even a malicious adversary knowing the algorithm
- legitimate stations have a secret that is not known by the adversary
  $\Rightarrow$ keyed MAC can be used to prevent faking messages by an adversary
- the adversary attempts to cause collisions so that the algorithm fails
- an adversary cannot use much higher communication resources than other users

Tricks - Cryptographic methods

- legitimate stations share a secret

Tricks - Cryptographic methods

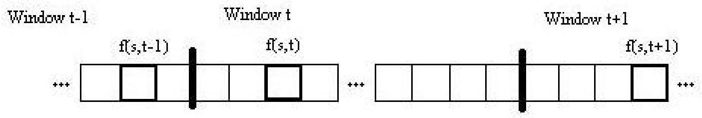
- legitimate stations share a secret
- all messages are enciphered and indistinguishable from a random noise

Tricks - Cryptographic methods

- legitimate stations share a secret
- all messages are enciphered and indistinguishable from a random noise
- the secret used to initiate a pseudorandom number generator
  each station can generate the same pseudorandom sequence

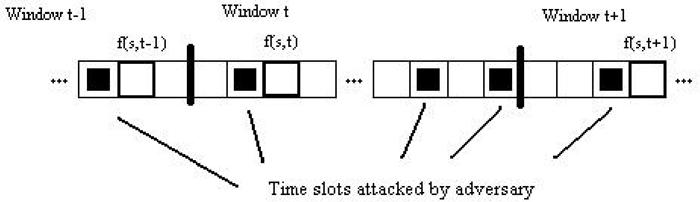
Tricks - Time windows

- within a group of steps only one used for communication
- which one is used depends on a pseudo-random value (f) computed from the secret (s) and current time (t)



Drawbacks of time windows

- limited immunity against an adversary
- increase of time



Tricks - Interleaving time windows

Technique used when groups of stations perform concurrently independent computations

- time window of length k used simultaneously by k groups
- for communication, group i uses slot

$$f(\mathrm{secret}, i, t)$$

f - cryptographic pseudorandom permutation

Tricks - Interleaving time windows

Technique used when groups of stations perform concurrently independent computations

- time window of length k used simultaneously by k groups
- for communication, group i uses slot

$$f(\mathrm{secret}, i, t)$$

f - cryptographic pseudorandom permutation

Advantages:

- each slot used – no waste of time
- from a point of view of a group – the same behavior as for time windows

Leader Election Algorithm (ESA'2003)

assumptions: a single-hop no-CD radio network with $\Theta(N)$ stations sharing a secret key. The stations are not initialized with ID's.

Leader Election Algorithm (ESA'2003)

assumptions a single-hop no-CD radio network with $\Theta(N)$ stations sharing a secret key. The stations are not initialized with ID's.

leader election algorithm energy cost – $O(\sqrt{\log N})$,
time complexity – $O(\log^3 N)$,
the outcome might be faulty with probability $O(2^{-\sqrt{\log N}})$ for an adversary with energy cost $O(\log N)$.

Leader Election Algorithm (ESA'2003)

assumptions a single-hop no-CD radio network with $\Theta(N)$ stations sharing a secret key. The stations are not initialized with ID's.

leader election algorithm energy cost – $O(\sqrt{\log N})$, time complexity – $O(\log^3 N)$, the outcome might be faulty with probability $O(2^{-\sqrt{\log N}})$ for an adversary with energy cost $O(\log N)$.

additional feature **it produces a group of $\Theta(\log N)$ numbered stations.**

Leader Election Algorithm (ESA'2003) - overview

$v = \Theta(\sqrt{\log N})$

- **Preprocessing** - we choose at random v small groups (each of size at most $O(\log N)$) of (pair of) candidates for the leader
- **Group elections** - *group election* phases executed in group 1, then in group 2, then ...

  The first group that succeeds in choosing a group leader "**attacks**" all subsequent group election phases preventing another leader to be chosen.

New Algorithm

Assumptions:

- a single-hop no-CD radio network consisting of $n = \Theta(N)$ stations
- stations share a secret key
- the stations are not initialized with any ID's, and are aware only of N

New Algorithm

Assumptions:

- a single-hop no-CD radio network consisting of $n = \Theta(N)$ stations
- stations share a secret key
- the stations are not initialized with any ID's, and are aware only of N

Solution features:

- energy cost $O(\sqrt{\log N})$
- time $O(N)$,
- the outcome is faulty with probability $O(2^{-\sqrt{\log N}})$ in a presence of an adversary with energy cost $O(\log N)$.

Initialization algorithm - overview

Idea: gradually increase the set of initialized stations

Initialization algorithm - overview

Idea: gradually increase the set of initialized stations

Phase 1: initialization performed concurrently in $k = \Theta(N/\log^3 N)$ groups of polylogarithmic size;

Initialization algorithm - overview

Idea: gradually increase the set of initialized stations

Phase 1: initialization performed concurrently in $k = \Theta(N/\log^3 N)$ groups of polylogarithmic size;

Phase 2: joining the groups from Phase 1 into a single set of $D = \Theta(n/\log^2 N)$ initialized stations whp;

Initialization algorithm - overview

Idea: gradually increase the set of initialized stations

- Phase 1: initialization performed concurrently in $k = \Theta(N/\log^3 N)$ groups of polylogarithmic size;
- Phase 2: joining the groups from Phase 1 into a single set of $D = \Theta(n/\log^2 N)$ initialized stations whp;
- Phase 3: 4 subphases, each of them increase the number of initialized stations by a factor of $\Theta(\sqrt{\log N})$ whp;

Initialization algorithm - overview

Idea: gradually increase the set of initialized stations

- Phase 1: initialization performed concurrently in $k = \Theta(N/\log^3 N)$ groups of polylogarithmic size;
- Phase 2: joining the groups from Phase 1 into a single set of $D = \Theta(n/\log^2 N)$ initialized stations whp;
- Phase 3: 4 subphases, each of them increase the number of initialized stations by a factor of $\Theta(\sqrt{\log N})$ whp;
- Phase 4: $\Omega(N)$ stations initialized; use them to initialize the remaining stations similarly as in Phase 3.

Phase 1: Initialization

- each station chooses independently a group from 1..k
- each group runs (modified) leader election (ESA'2003)
  – with interleaving
- result: $\Theta(\log^3 N)$ groups of $\Theta(\log N)$processors.

Phase 1: Initialization

- each station chooses independently a group from 1..k
- each group runs (modified) leader election (ESA'2003) – with interleaving
- result: $\Theta(\log^3 N)$ groups of $\Theta(\log N)$processors.

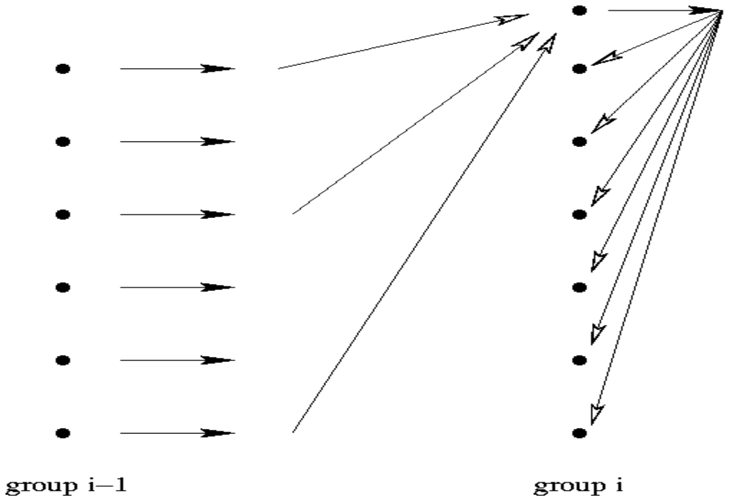
What can an adversary do:

- attack at most $O(\log N)$ groups
- even attacking a single group difficult – leader election is adversary immune!

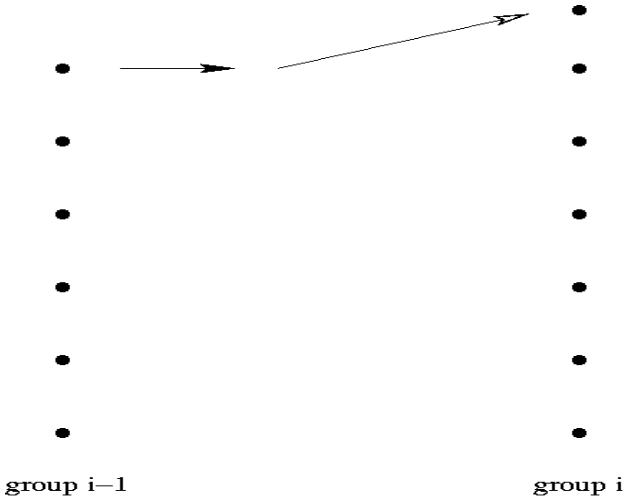
Phase 2: Joining initialized sets

- counting the number of initialized stations
- group i gets a number x of initialized stations in groups 1 through $i-1$ and initializes its stations with
  $x+1, x+2, \ldots,$
  and informs group $i+1$
- if something goes wrong – group i discarded

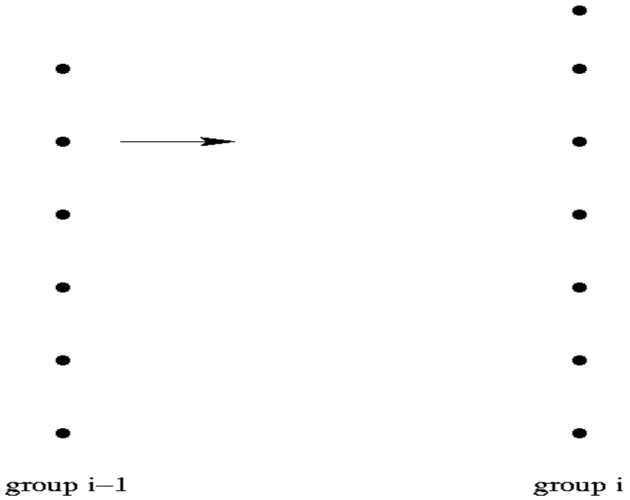
Phase 2: the whole communication pattern



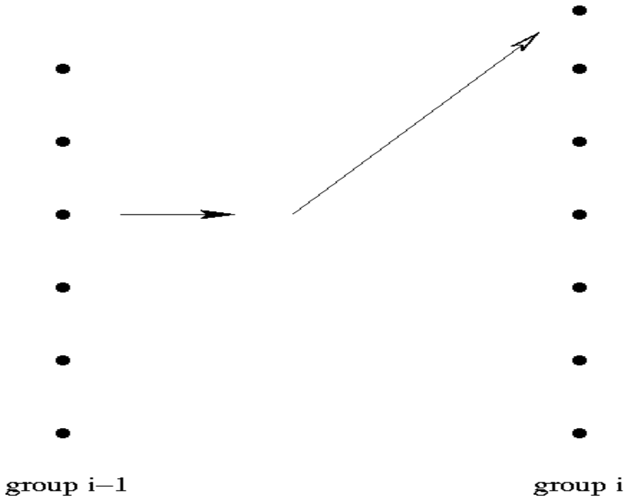
Phase 2: step 1



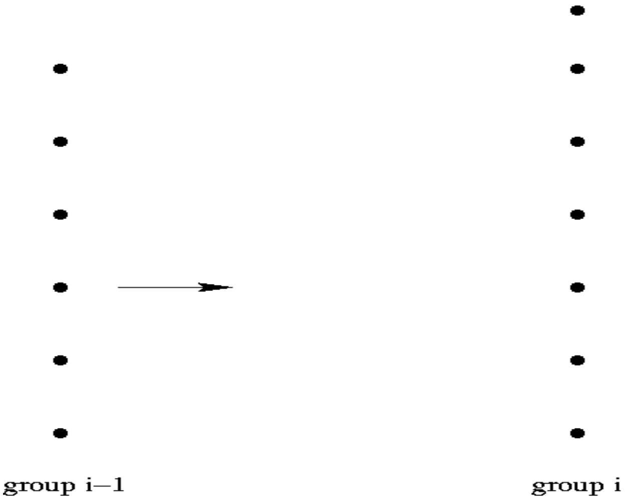
Phase 2: step 2



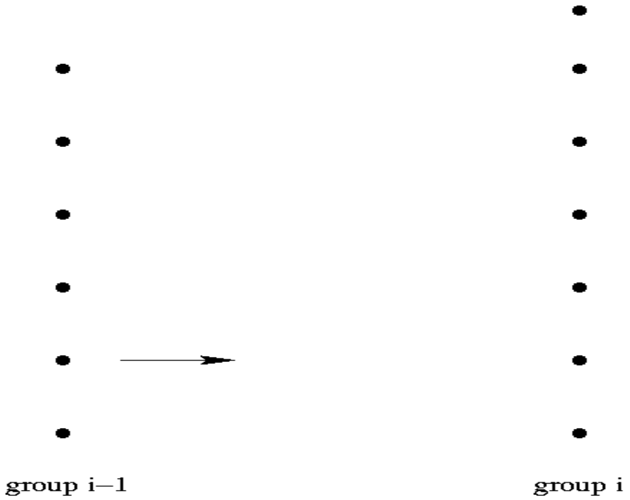
Phase 2: step 3



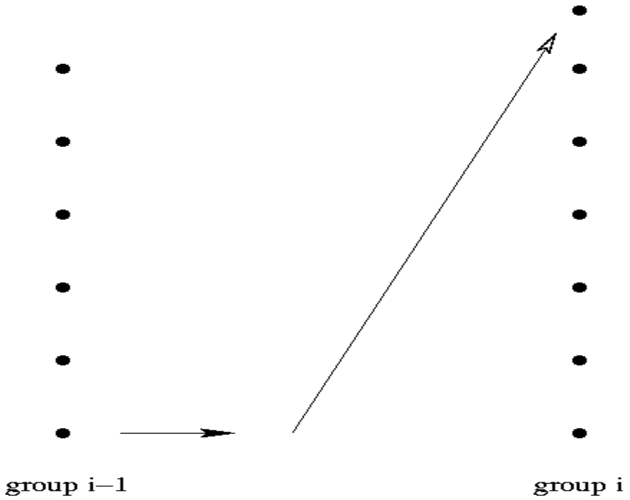
Phase 2: step 4



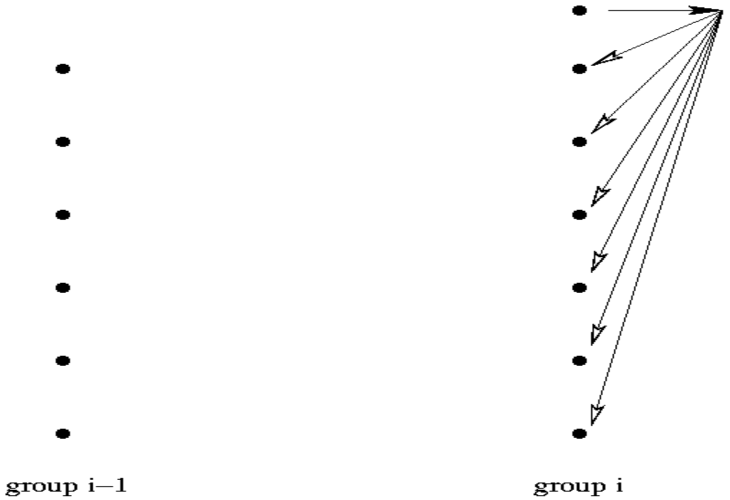
Phase 2: step 5



Phase 2: step *r*



Phase 2: broadcast to group *i*



Phase 2: adversary

- the adversary may cause discarding a small number of groups
- but **he cannot make the computation inconsistent**

Phase 3:

Overview:

3a already initialized stations split into *collection groups*,
each groups collects yet uninitialized stations

3b collection groups merged – similarly as in Phase 2

Phase 3a -overview

- some G collection groups formed
- each collection group has a number of auxiliary stations – *servants* that mantain communication there

Phase 3a -overview

- some G collection groups formed
- each collection group has a number of auxiliary stations – *servants* that mantain communication there
- uninitialized stations choose a collection group and step

Phase 3a -overview

- some G collection groups formed
- each collection group has a number of auxiliary stations – *servants* that mantain communication there
- uninitialized stations choose a collection group and step
- inside a group: relay procedure used to collect some number of uninitialized stations that have chosen this group

Relay procedure:

step t of a collection group:

- a servant informs about the number of stations collected so far
- each uninitilaized station that has chosen this group and t responds
- if no collision (SINGLE message), the servant sends an acknowledgment

Relay procedure:

step t of a collection group:

- a servant informs about the number of stations collected so far
- each uninitilaized station that has chosen this group and t responds
- if no collision (SINGLE message), the servant sends an acknowledgment

Design problems:

- a servant used only $O(\sqrt{\log n})$ times
- fine design of switching the roles between the servants
- an adversary cannot cause inconsistencies – even if some of the messages get scrumbled
  what happens if the acknowledgment comes not through?

Remarks and conlusions

- if the adversary detects an encoded transmission – to late for collision, our techniques still work
- multihop networks
- small network sizes: as always – a combination of the same tricks but tuned for the size of n (e.g. $\sqrt{n}$ might be smaller than $\log^2 n$)

Alexander von Humboldt
Stiftung / Foundation

The speaker's attendance
at this conference was sponsored
by the
Alexander von Humboldt Foundation.

- http://www.humboldt-foundation.de