Generalizing Mixes

Claudia Diaz
Andrei Serjantov

PET Workshop 26/03/2003

Outline

- What is a mix?
- Batching strategies
- Generalizing mixes: concept and examples
- Anonymity set size
- A new mix function
- Randomizing mixes: the Binomial mix
- Properties of the Binomial mix
- Conclusions and future work

Introduction

- What is a mix (Chaum81)?
  - Building block of anonymous communication systems
  - Router that hides correspondence between incoming and outgoing messages
  - Collects messages together, “mixes them up” and forwards them on

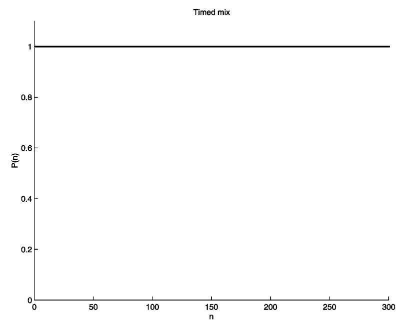
Batching strategies

- Timed / Threshold / Combination
  - Flushing: timeout and/or number of messages?
- No pool / pool / dynamic pool (Cortrell)
  - Does the mix keep some messages for the next round? Which criteria is used?
- Stop-and-go (Kesdogan98): a different concept
- Tradeoff anonymity / message delay

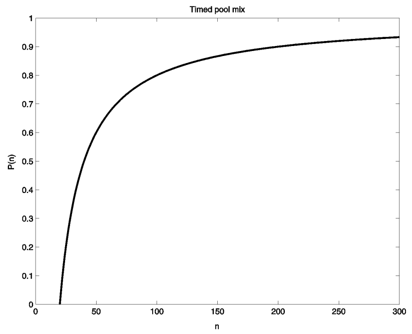
Generalizing mixes: the concept

- So far, mixes were described by the algorithm
- We represent the mix as a function P from the number of messages inside the mix to the probability of a message of being sent
- The function represents the mix at the time of flushing

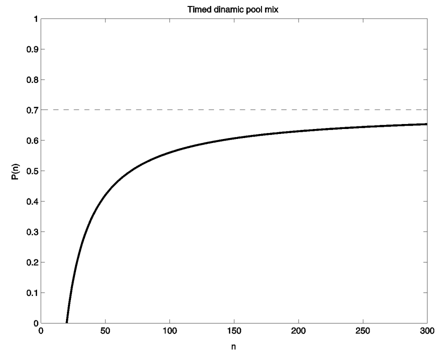
Example 1: Timed mix



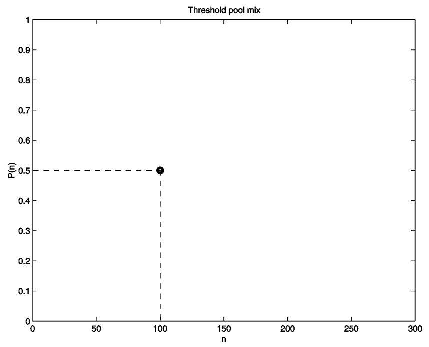
Example 2: Timed pool mix



Example 3: timed dynamic pool mix



Example 4: Threshold pool mix



Anonymity set size

- Anonymity: "*state of being not identifiable within a set of subjects, the anonymity set*" (Pfitzmann and Köhntopp)
- The *effective* size of the anonymity set can be computed using the **entropy** of the probability distribution that relates incoming and outgoing messages

Anonymity set size

- It depends on two parameters
  - Number of messages mixed
  - Distribution of probabilities
    - A *priori* or contextual information
    - Probability of a message leaving in each round (pool mixes): the more evenly distributed the probability of a message leaving in round *r*, the more anonymity

Anonymity set size

Probability of and output matching an input:

- n_r: number of messages inside the mix at round *r*
- $P(n_r)$: probability of a message leaving at round *r* (represented function)
- prob(i): probability of an outgoing message (round *r*) matching an input at round *i*

$$prob(i) = P(n_r) \prod_{j=i}^{r-1} (1-P(n_j))$$

Note: when the $P(n_j)$ are high, the prob(i) are less evenly distributed => lower entropy, less anonymity, less delay

Tradeoff: anonymity / delay

- Given the two parameters that determine anonymity (number of messages mixed and distribution of probabilities):
  - If there are many messages, we can improve the delay (we can allow less evenly distributed prob(i) -> higher $P(n_j)$)
  - If there are few messages we should guarantee sufficient anonymity (lower $P(n_j)$)

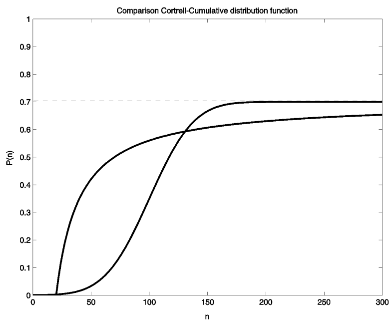
Proposed mix function

- Using the graphical generalization of mixes, we can easily define an arbitrary P(n) function that satisfies particular requirements as:
  - Tolerated delay
  - Minimum anonymity set size
- Depending on the parameters of the system as:
  - Number of users
  - Traffic load

Proposed mix function

- We propose normal cumulative distribution function
- Properties:
  - Improved anonymity in low traffic conditions
  - Improved delay in heavy traffic conditions
  - Smooth growth

Proposed mix function: comparison to Cortrell's mix (qualitative example)



Randomizing mixes: the Binomial Mix

- P(n) determined in previous designs the fraction of messages flushed. Now, we take it as a probability
- For each message we toss a biased coin and send it with probability *p=P(n)*
- The number of sent messages, *s*, follows a binomial distribution (on average *s=nP(n),* the variance is *np(1-p)*)
- Effect: the mix hides the number of messages contained in it, *n*

Guessing the number of messages contained in the mix

- The attacker has to guess *n* by observing *s* (applying Bayes' rule)
- Practical results (simulator):
  - One observation of the output (average values):
    - The attacker guesses n with probability ~1%
    - In the 95% confidence interval lay ~17 values
    - The average entropy is 4.17 bits
  - Number of rounds of observation needed to estimate n with 95% confidence ~125
- Not practical to do it this way

The blending or *n-1* attack

- In classical mixes: the attacker empties the mix of unknown messages, fills it with his messages and then lets the target message in: the target message has no anonymity
- In the Binomial mix the attacker does not know how many messages are in the mix: how does he know when it is empty of unknown messages?

Advantages of the binomial mix

- The success of the blending attack is probabilistic
- The attacker has to make a greater effort to attack the mix
- If a dummy traffic policy is implemented, the fact of hiding the number of messages makes blending attacks harder

Conclusions

- We propose a framework with which we can generalize classical pool mixes
- The model gives us a new understanding of classical batching strategies
- New batching strategies arise from the framework. We can have a tailored anonymity/delay tradeoff that adapts to fluctuations in the traffic load
- We suggest a new an intuitive way of dealing with the anonymity set size, as a function of P(n)
- We have added randomness to the mix in order to increase the effort of the attacker when deploying blending attack. The success of this attack becomes probabilistic

Future work

- Further analysis of the possibilities of the framework. Search other functions that may have interesting properties
- Thorough analysis of the binomial mix and the implications of hiding the number of messages contained in it
- Study the properties of the binomial mix when a dummy traffic policy is implemented

Questions ?

The blending attack in the Binomial mix

- **Attack model:** global, active, external attacker; can delay and insert messages
- **Emptying phase**: the attacker floods the mix in order to increase $p=P(n)$ to $p_{max}=P(N_T)$. The success is **probabilistic** ($1-\varepsilon$), a function of the number of 'flooding' rounds
- Number of rounds r: $(1-(1-p_{max})^r)^n > 1-\varepsilon$

Note: the attacker has to assume worst case: $n=N_{max}$

Effort of the attacker to empty the mix

- Number of messages the attacker has to send to the mix: $N_T+(r-1)(N_T+n)p_{max}$
- Time needed (T is the *timeout* of the mix): rT
- Number of messages the attacker has to delay (assuming Poisson traffic with average λ): λrT
- The flushing phase is similar to previous mixes