Privacy Preserving Business Process Fusion

Roberto Guanciale and Dilian Gurov, KTH Sweden

supported by UaESMC EU Project

SBP - 2014-09-08

Virtual enterprises

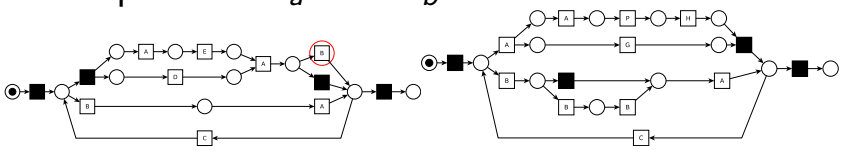
- temporary and loosely coupled alliances of businesses
- possibly competitive parties
- supported by IT systems

Process fusion - informal

- let a and b be two enterprises building a VE
- the two enterprises own a "secret" business process (N_a and N_b)
- the VE behavior is driven by the "composition" of the participant processes
- the participants can log the "shared" events
- if the VE persists, the participants can learn the set of all possible traces of "shared" events
- a-priory knowledge of this information can support optimizations

Process fusion - formal

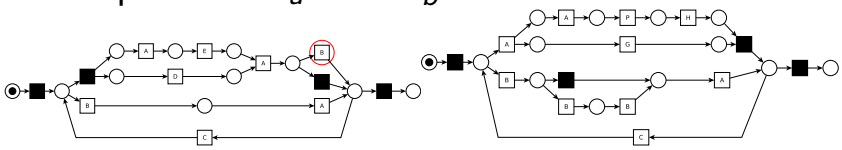
- a and b be own two bounded labeled Petri nets N_a and N_b over alphabets Σ_a and Σ_b



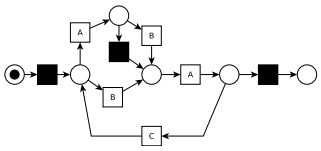
- the VE behavior is modeled by $N_a \times N_b$
- $F_i(N_a, N_b)$ computes N_i' such that $N_i' \sim \mathbf{proj}_{\Sigma_i}(N_a \times N_b)$

Process fusion - privacy

- a and b be own two bounded labeled Petri nets N_a and N_b over alphabets Σ_a and Σ_b



- If $\mathbf{proj}_{\Sigma_a}(N_a \times N_b) \sim \mathbf{proj}_{\Sigma_a}(N_a \times N'_b)$ and $\Sigma_a \cap \Sigma_b = \Sigma_a \cap \Sigma'_b$ then $F_a(N_a, N_b)$ is indistinguishable from $F_a(N_a, N'_b)$



- no trusted third party

Background

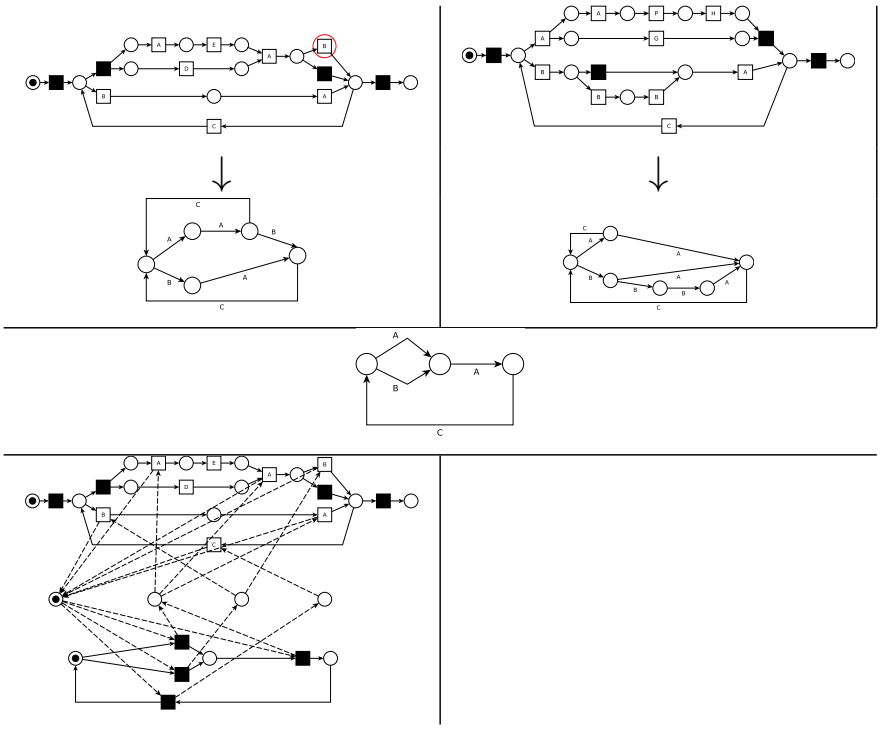
- Let A_1 and A_2 two DFA over the common alphabet Σ
- $SMC_{\times}(A_1, A_2)$
- Secure Multiparty Protocol to privately compute $A \sim A_1 \times A_2$
  - additive secret sharing
  - automaton minimization
  - implemented in Sharemind
  - uses three un-trusted third parties
  - secure against passive adversary (corruption up to two parties)

Protocol

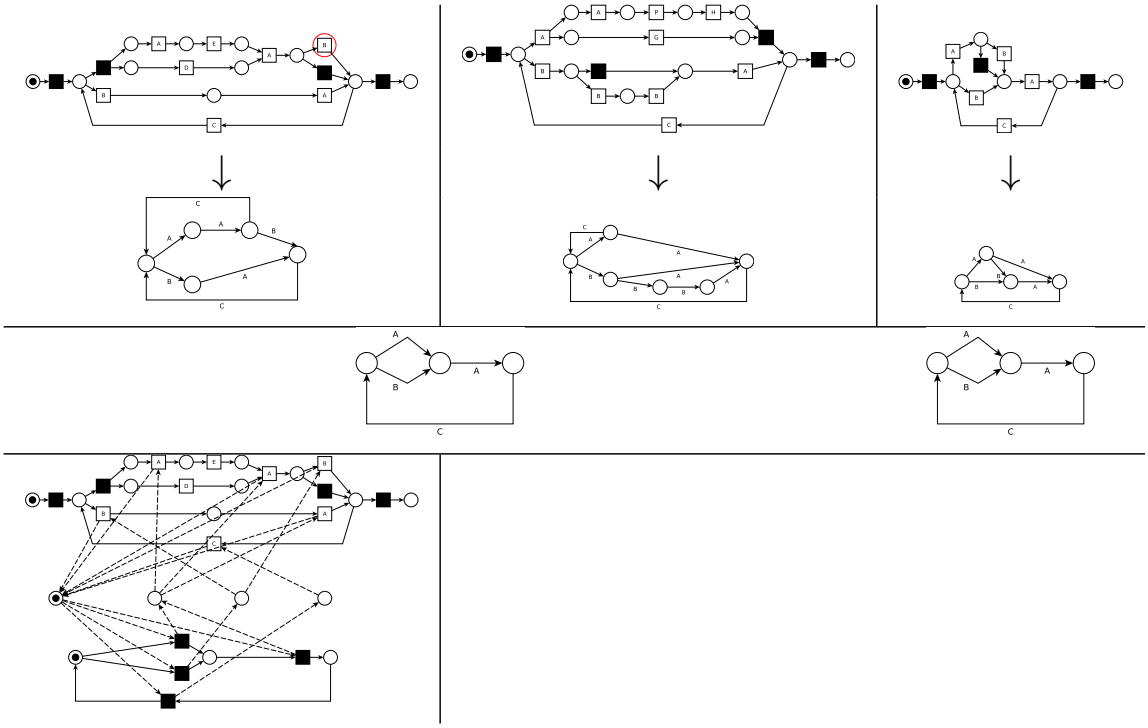
Algorithm 1: $F_i(N_a, N_b)$

1. $N_i^p := \mathbf{proj}_{\Sigma_a \cap \Sigma_b}(N_i)$ // project the private net on the // common alphabet
2. $A_i^p := NFA(N_i^p)$ // obtain the equivalent NFA // by computing the reachability graph
3. $A_i^d := Det(A_i^p)$ // determinize the automaton
4. send A_i^d to the secure multiparty protocol
5. receive $A := SMC_{\times}(A_a^d, A_b^d)$
6. $N := Reg(A)$ // synthesize the corresponding Petri net
7. return $N_i \times N$ // apply the constraints to the initial net

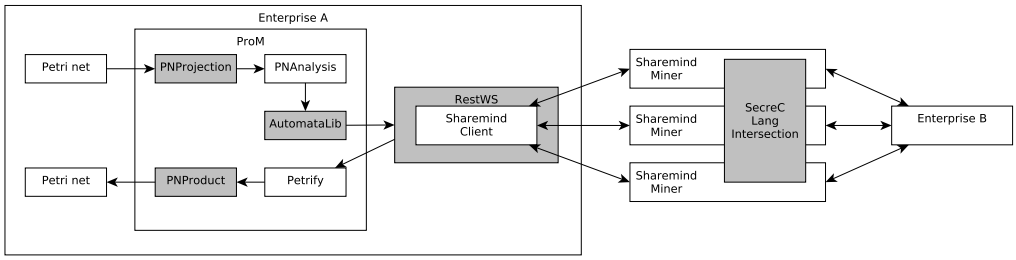
Example



Example



Implementation



- based on ProM and Sharemind
- execution time dominated by the SMC protocol
- 10 states, 4 labels $\Rightarrow$ 185 seconds
- 100 states, 20 labels $\Rightarrow \sim$ 15 hours

Ongoing research

- process fusion is based on the notion of trace-equivalence
  - easy to justify the privacy properties
- trace equivalence do not preserve deadlocks
  - the result can not be used to locally check if the VE process is sound

Ongoing research

- move to different composition operators
  - e.g. $\oplus$ of Open nets
- focus on weak-termination
- existing solutions handle to top-down approach
- use SMC to handle the bottom-up