Formally verified confidentiality guarantees for a Blinded Memory SoC model

- **BliMe**: confidential outsourced computation from hardware information flow tracking
- **Challenge**: a BliMe SoC must track information flows across many components
- **Solution**: formal model with trusted/untrusted peripherals ⇒ formally-verified confidentiality

**Blinded Memory (BliMe) [1]**
- **Attestation** to assure client the system is enforcing BliMe architecture
- **Encryption engine** to decrypt+taint and untaint+encrypt client data
- Hardware enforced **taint tracking policy** to prevent tainted data to leave the system

![Architecture of a BliMe system](image)

**Extension to BliMe model**
- A **multi-peripheral** model of System
- A **peripheral firewall** to ensure safety in the presence of untrusted peripherals
- Fixing the safety definition to include **inter-client confidentiality violations**
  - Old: ∀ s1, s2 ∈ S : s1 ≡ s2 ⇒ X(s1) ≡ X(s2)
  - New: ∀ s1, s2 ∈ S and d ∈ D : s1 ≡ s2 ⇒ X(s1) ≠ X(s2) and Leakage(X(s1)) = Leakage(X(s2))

**The problem**
- The initial BliMe model has a **single CPU** with direct access to memory and accelerator as a blocking CPU instruction
- Not all peripherals enforce security policy

**References**

**Conclusion**
- Increasing the confidence in BliMe
- Extending BliMe with peripheral firewall
- Future direction: Extracting synthesizable hardware design from model with formally verified properties

**Acknowledgments**
This work is supported in part by the Academy of Finland (decision 339514).