Synthesizing Safe Timing Bounds for Secure CAN bus in Automotive Applications

ABSTRACT

abstract here

ACM Reference Format:
. 2018. Synthesizing Safe Timing Bounds for Secure CAN bus in Automotive Applications. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, **??** pages. https://doi.org/10.475/123_4

1 INTRODUCTION

Timing side channel attacks can leak information from an otherwise secure authentication protocol. In a typical functional correctness verification setting, it is sufficient to build a platform independent model of the protocol, and check the desired properties against that model. While this methods ensures a protocol has the correct functional behavior, it cannot capture the timing behaviors guarantees needed for side channel safety. Any guarantees made on a platform independent model of the protocol might be violated when it is implemented on a specific platform which introduces further delays. As such, we use a platform dependent, parameterized timed automata model of the protocol and synthesize safe delay bounds that will ensure timing behaviors are preserved. We demonstrate feasibility of our approach by synthesizing delay bounds for the MaCAN authentication protocol.

Ideally we would like the following workflow.

(1) A protocol designer comes up with a platform independent model of their protocol
(2) Given a timing side channel safety property expressed in HyperLTL, we synthesize constraints on the delays that a platform specific implementation is allowed to introduce on each transition such that the property still holds.
(3) the protocol, along with the implementation bounds, is given to a user. The user, who has a bunch of black boxes with worst case execution times for each box, is free to combine black boxes in whichever is needed as long as the combination still satisfies the given bounds.

1.1 Adding a time buffer

The easiest way to ensure there are no side channel timing attacks is to add a dynamic buffer to every transition so that the observable output always occurs at the same time. There are two drawbacks to this.

The first is that this necessarily slows everything down to the slowest common denominator. So we might build a system in an inoptimal way. **Mark:** [not sure about this - gotta formalize is].

The second is that this requires an extra step that is really, not necessarily, necessary. Instead of adding extra parts, we would like to just use the components we are given, in the way the were intended to be used. So if we have all these black boxes, I would rather combine them together so that they are timing safe, rather than slap them together any old way and then have to add delay boxes all over the place.

REFERENCES

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 123-4567-24-567/08/06.
https://doi.org/10.475/123_4