# Time-Based Link Layer Authentication Simulation with Elixir / Erlang

Ian Johnson

## 1  The Time-Based Authentication Protocol

The simulator described in this report is used to collect data pertaining to the efficacy of the Time-Based Authentication (TBA) protocol. The TBA protocol is a link-layer authentication protocol which operates under the assumption that a user can send a message whenever it becomes available on their system (this exists in a CDMA network, for example). Both parties in a TBA-authenticated system run parallel stream ciphers, which they initialize using a pre-existing shared secret, or through a Diffie-Hellman key exchange. The two parties use the parallel ciphers to encode a shared secret delay value. Whenever a party has a message to send to the other, it induces an artificial delay based on the most recent output of the stream cipher. Upon receiving the message, the other party measures the total response time since their prior message, and ensures that the correct amount of time was induced. The protocol has specifications for setup, tear-down, error recovery, and threat response, all of which are explained in the appendix of this report.

The remainder of this report will consist of a description of the simulation tool used to test this proposed protocol, as well as some initial results from running the simulator.

## 2  Simulating the TBA Protocol

The simulator for the TBA protocol is a parallelized Elixir program which can run on any arbitrary number of separate processes, processor cores, or nodes on a server cluster. However, note that running the simulator on a network-connected cluster which does not provide constant message delay violates the simulator's assumption that there is negligible propagation time for messages between nodes.

### 2.1  Elixir/Erlang Background

Erlang is a VM-based functional programming language with strong, deep-seeded support for parallelization and distributed programming. Elixir is, in essence, a modern syntax and library set for the Erlang language. A few nomenclature distinctions must be made with respect to the two languages before the simulator itself can be discussed.

First, the (simplified) architecture of the Elixir/Erlang environment must be known. Erlang runs a VM called Beam. The Beam VM can contain multiple Erlang processes. Erlang processes have very low overhead, so millions of them can run on a single Beam instance at once. However, the Beam VM runs on a single operating system-level process. Therefore, a simple parallelized program running in a single Beam VM is not truly parallel, as all computations are occurring on the same physical CPU core. However, Erlang processes running on Beam are all isolated from one another, and can send eachother messages based on their Beam process IDs (PIDs).

1

Therefore, they can, with reasonable accuracy, be used to simulate truly parallel processes with neglibile message propagation delay.

The data presented in this report was generated using parallel Erlang processes running in a single Beam VM on a single CPU core. However, the simulator itself is configurable to use any number of cores, on any number of machines.

## 2.2   Simulator Architecture

## 2.3   Using the Simulator

# References

[1] here is the citation