



Secure History Preservation Through Timeline Entanglement

Petros Maniatis, Mary Baker
<http://identiscape.stanford.edu/>

*Time and history are important concepts in a distributed system, especially one whose participants are highly likely to misbehave or cheat. We define a **Secure Timeline**, a tamper-evident record of the states traversed by a system in its history, based on one-way hashing and secure time stamping. We also propose a technique called **Timeline Entanglement** that allows the correlation of distinct secure timelines, maintained by independent, mutually distrustful systems. Timeline Entanglement is a sound, efficient, scalable, and survivable mechanism to protect the historic integrity of a distributed system.*

Who cares about history?

- Prior art issues
"Who published invention A first?"
- Historic file systems
 VMS, Plan 9/Venti, Elephant
- Reputation-based systems
"When did the feedback get posted?"
- Accountable systems
 If system answers X to question Q, it should be held to it

Target Setting

- Dynamic group of systems
- Malicious behavior is **common**
- No globally trusted arbitrator

Why isn't a log enough?

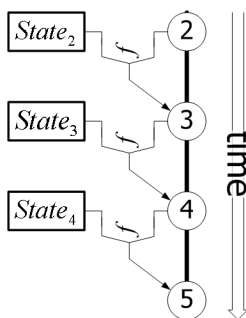
- The log can be modified and no one will be able to tell
- Log entries may claim a time when each was produced, but those time stamps can also be modified clandestinely

What about "agreement"?

- A lot of theoretical work has been done on achieving agreement in the face of malice
- Most resulting algorithms
 - Are very complex
 - Are very slow
 - Require bounds on how much malice exists
 - Do not behave well in highly dynamic settings

The Secure Timeline

- An authenticated, historic logical clock within a single system
- Relies on the one-way property of cryptographic hash functions, such as SHA-1
- Latest authenticator is a fingerprint of the entire timeline

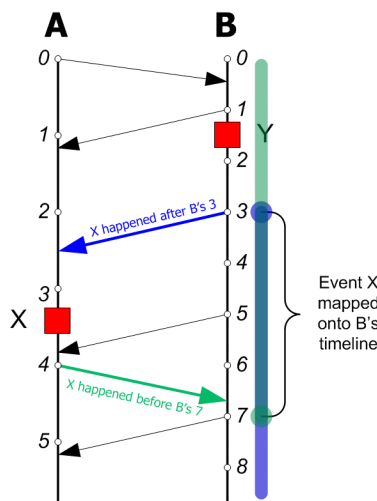


Timeline Entanglement

- Used within a dynamic group of independent systems
- Each system maintains its own secure timeline
- Periodically, each sends its current authenticator, called a **thread**, to all other members
- Threads received are archived and included in the receiver's system state
- Receiver's next authenticator is derived from the sender's timeline authenticator
- Entanglement links the past of the sender to the future of the receiver

Temporal Mapping

- B wishes to compare temporally its own event Y with A's event X, which happened between $\langle A,3 \rangle$ and $\langle A,4 \rangle$
- In the example, the thread from $\langle B,3 \rangle$ to A verifies that event X happened after $\langle B,3 \rangle$, in interval $(3, \infty)$
- Similarly, the thread from $\langle A,4 \rangle$ to B verifies that event X happened before $\langle B,7 \rangle$, in interval $[0,7)$
- Consequently, the mapping of event X onto B's timeline is the intersection of the two intervals, $(3,7)$
- This mapping is undeniable, regardless of whether A and B trust each other or not, because of the one-way property of the hash function



Properties

- Entanglement is sound but not complete
 - Proved precedences are correct, but
 - Not all precedences are provable
- Historic integrity is preserved
 - Every node acts as a watchdog for every other node's historic integrity
- Historic proofs are survivable
 - The maintainer of a timeline need not participate during proof verification

What does this cost?

- Setting:
 - 1 time step per second
 - Nodes entangle every 1 hour
- CPU and I/O per time step
 - 50ms for group size of 3600
 - 400ms for group size of 28800
- Data sent per time step
 - 2 Kbytes/sec for group size of 3600
 - 20 Kbytes/sec for group size of 28800
- Storage requirements
 - ~10 Gbytes per year (and improving)

Is there a catch?

- All-to-all entanglement is expensive for large groups **and** fine granularity
 - For 10-minute entanglement, 1200 nodes spend 10% of their time entangling
 - Can be improved by using less dense entanglement graphs
 - Use a dynamic overlay algorithm (e.g., CAN) and only entangle between direct neighbors
- Mapping comes with a loss in resolution
 - For 10-minute entanglement, mapping "spreads" one time step between 10 and 20 minutes
 - When mapping between remote nodes, pick the route that minimizes aggregate temporal loss
- Naughty peers might report different authenticators to different neighbors
 - Use reliable or consistent broadcast to verify delivery of threads by everyone
 - Corroborate a reported thread by comparing to those received by other members in the group

