Securing Time in Untrusted Operating Systems with TimeSeal

Fatima M. Anwar, Luis Garcia, Xi Han, Mani Srivastava

Vulnerabilities in Time

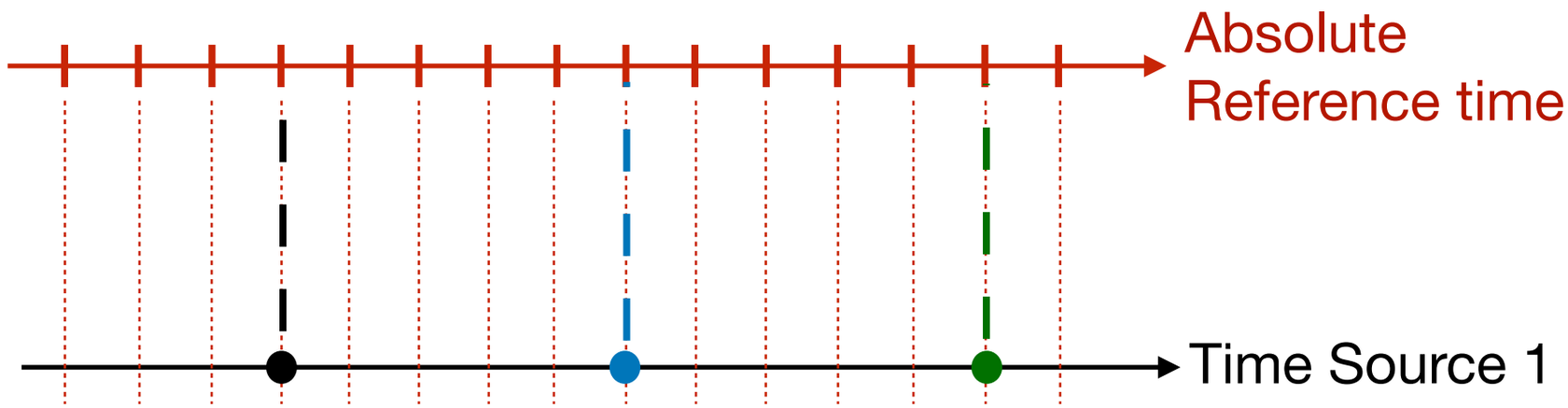
Causes

- Malicious *delays*
- Inconsistent *rates*
- Induced *drifts*
- Non monotonic *timers*

Consequence

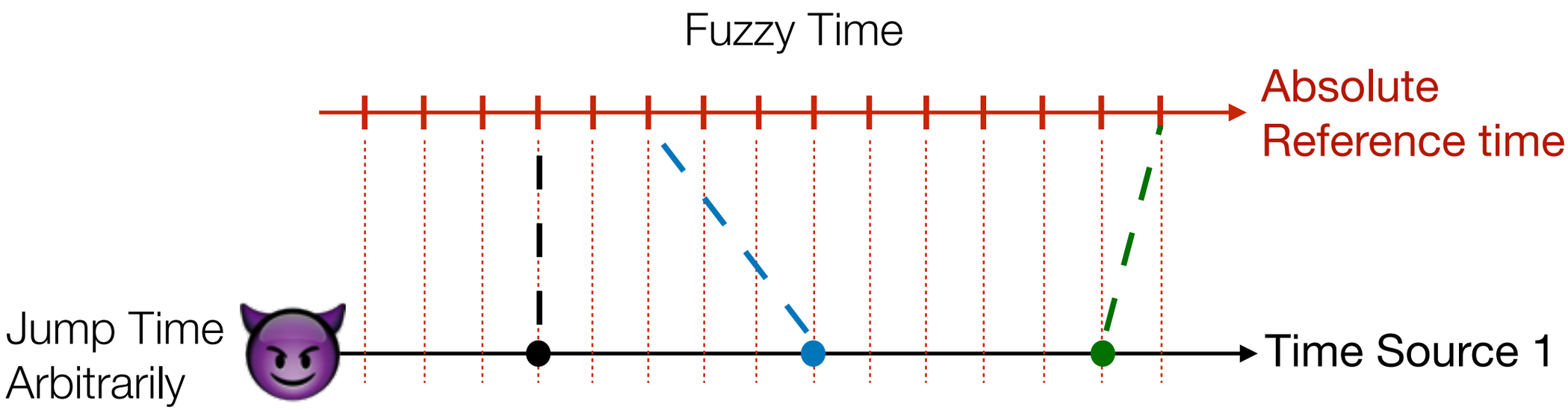
Time discontinuities,
Fuzzy Time

Attack Strategies



No Attack

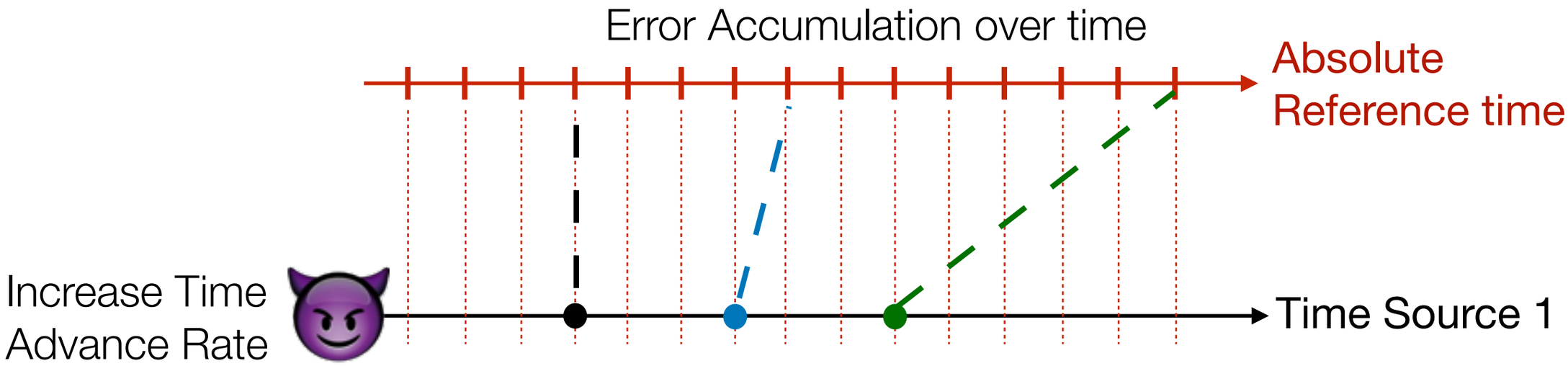
Attack Strategies (1)



Offset Manipulation

Attack Strategies (2)

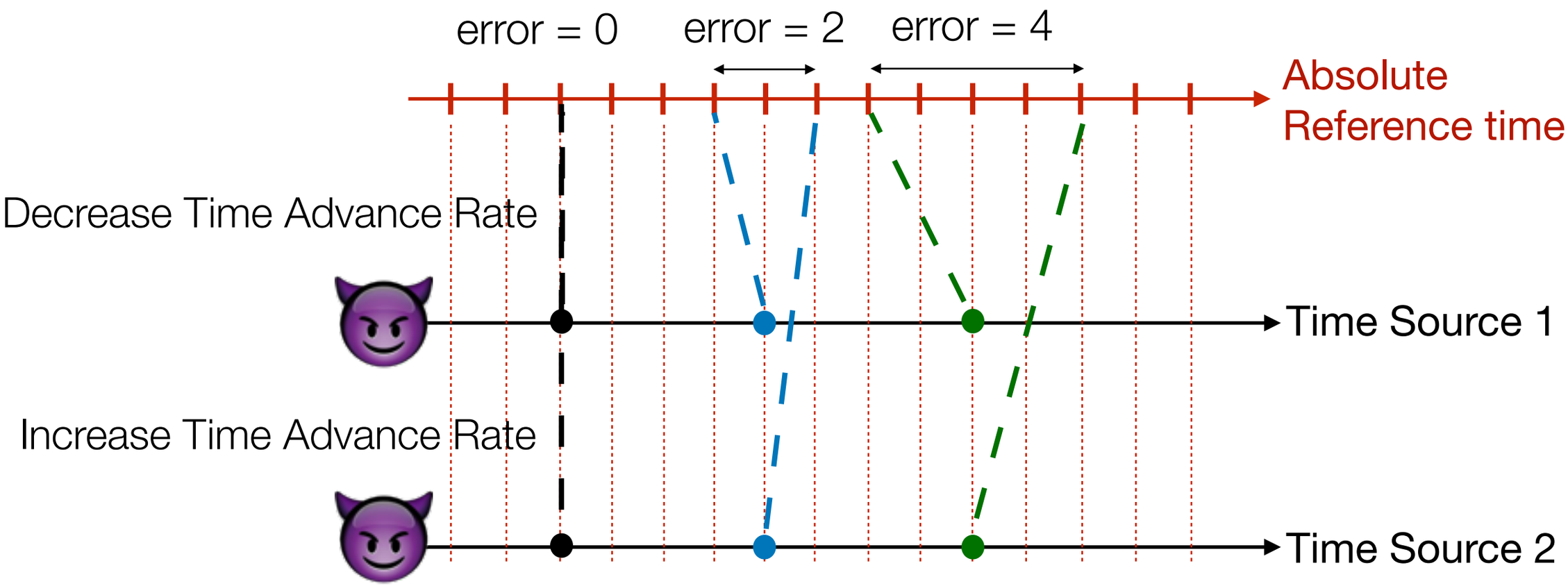
Effect on a Single Time Source:



Rate Manipulation

Attack Strategies (2)

Relative Effect on Two Time Sources:



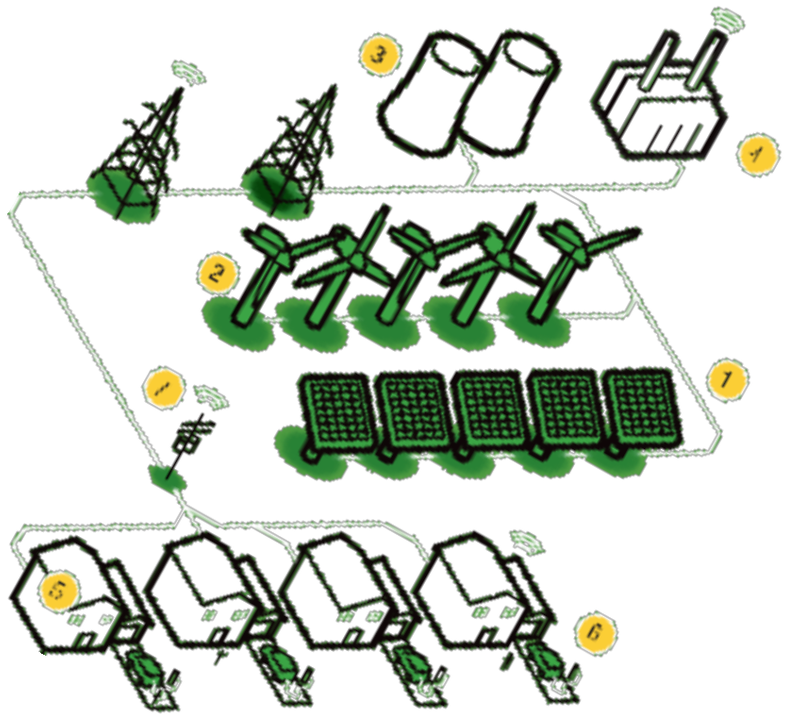
Rate Manipulation

Time Attack Consequences

Forge Timestamp

Location theft

Grid Attack



Violate DRM

Time Attack Consequences

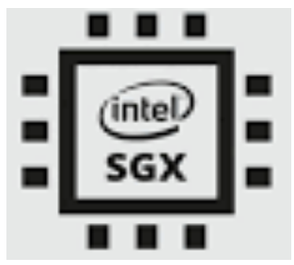
Forge Timestamp

Location theft

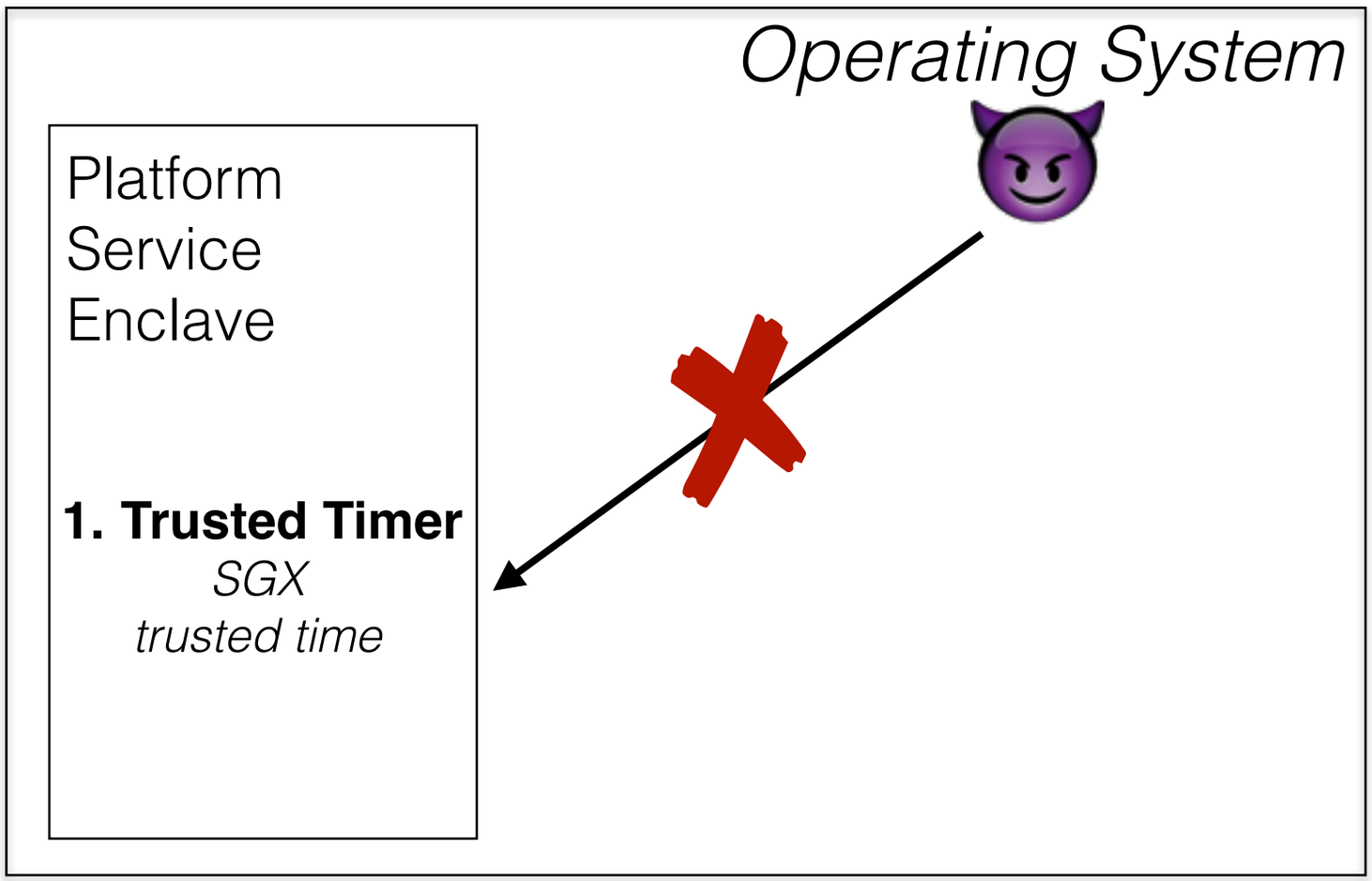
Do Trusted Platforms provide Secure Time?

Securing Time Challenges

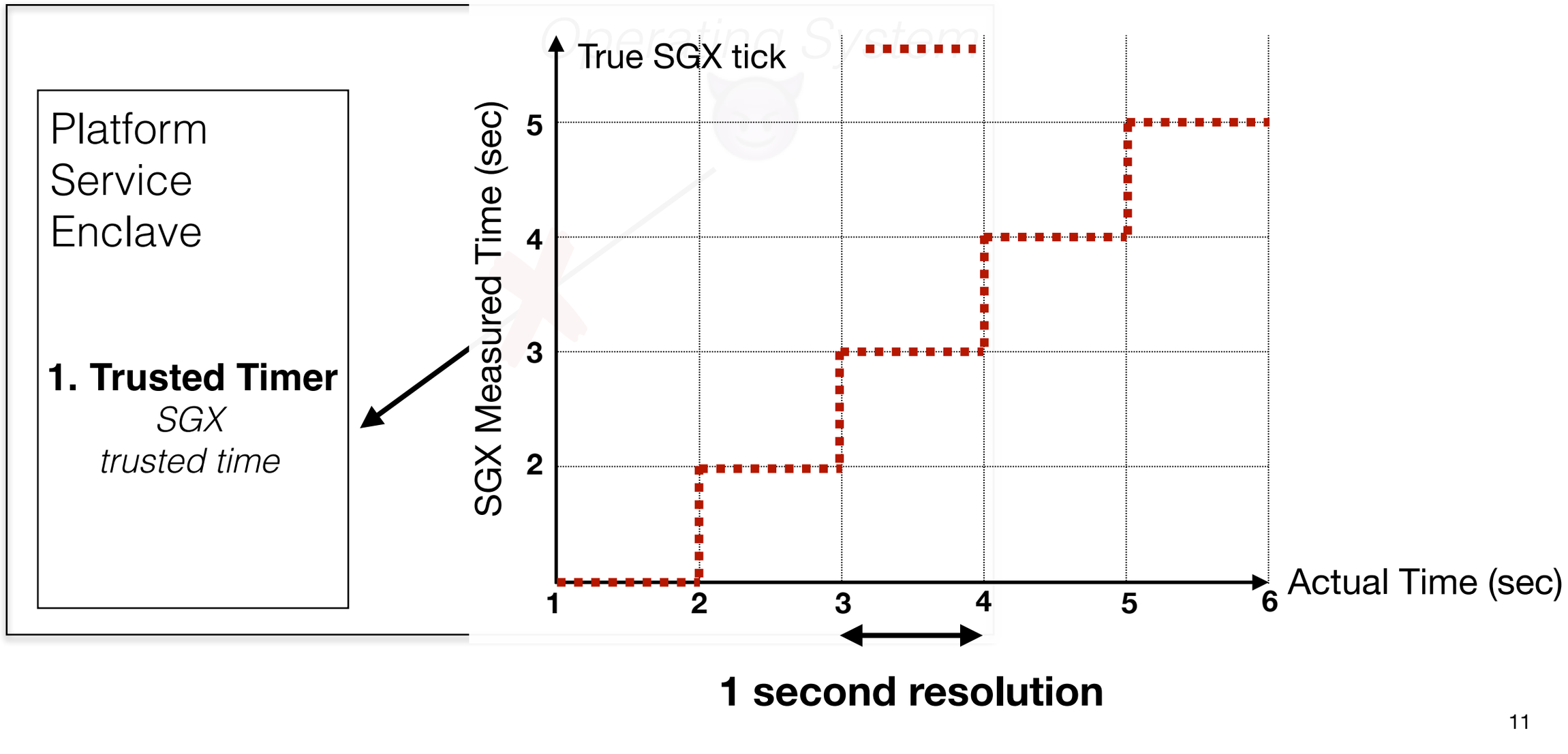
1. A Trusted Timer
2. Secure Access to the Trusted Timer
3. Secure Timekeeping Software
4. Secure Access to Global Time on the Network



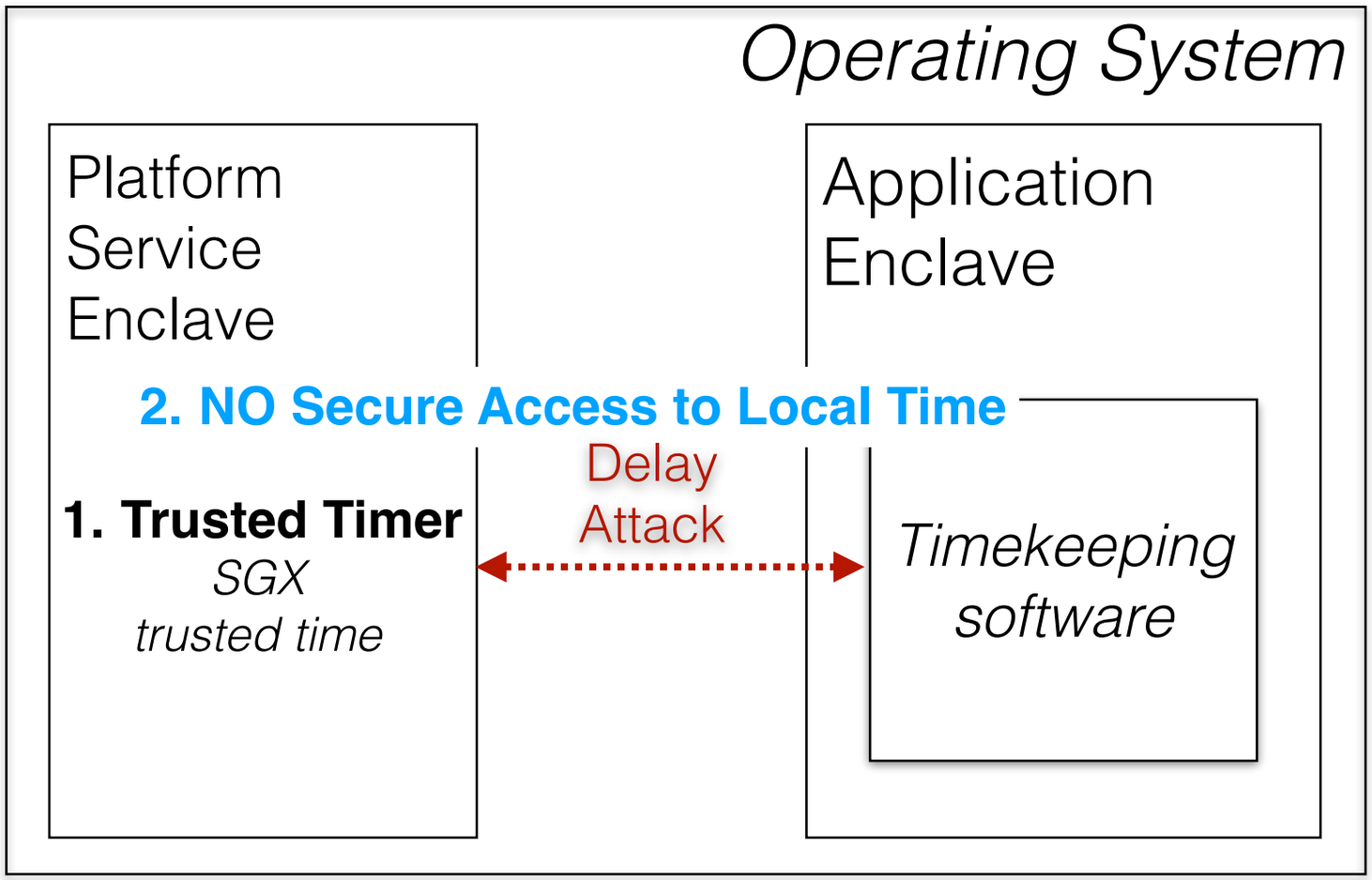
SGX “Trusted” Time



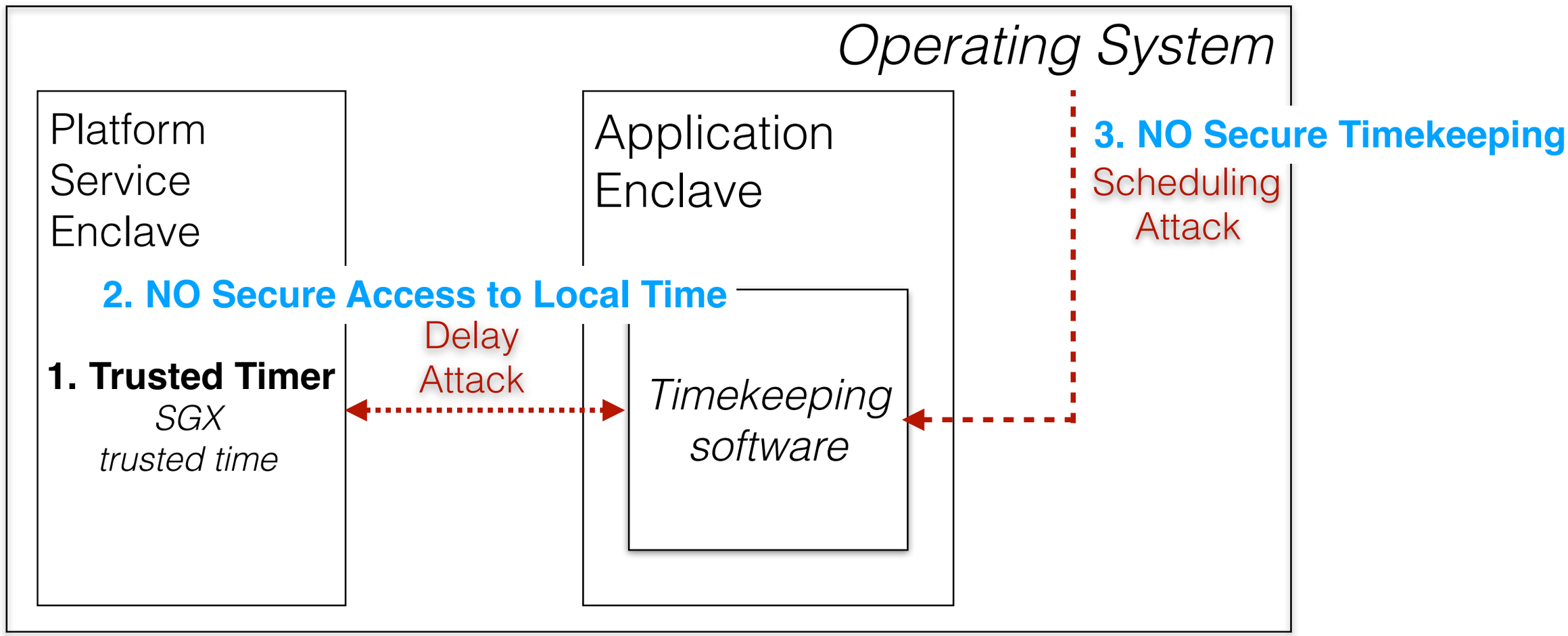
SGX “Trusted” Time



Attack on SGX “Trusted” Time

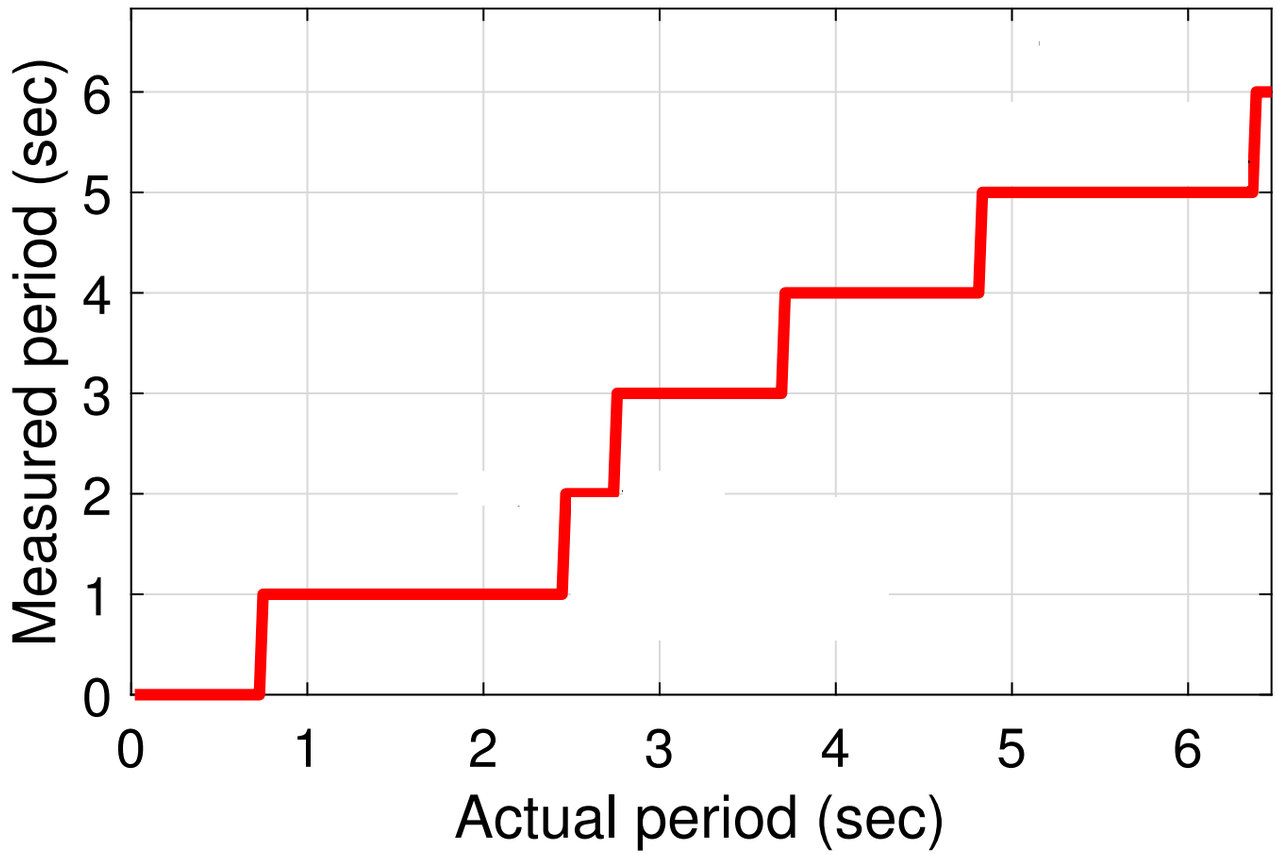


Attack on SGX “Trusted” Time



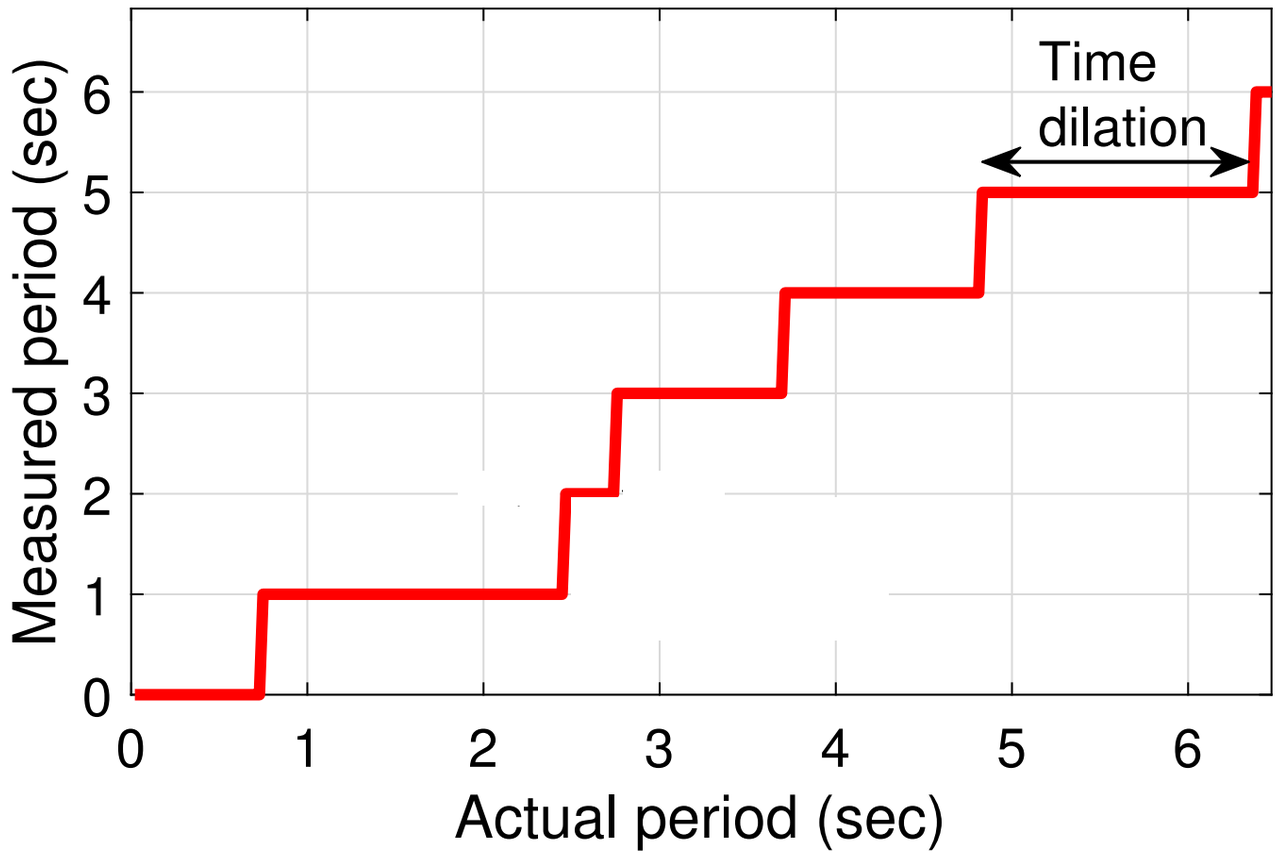
Attack on SGX “Trusted” Time

Result:



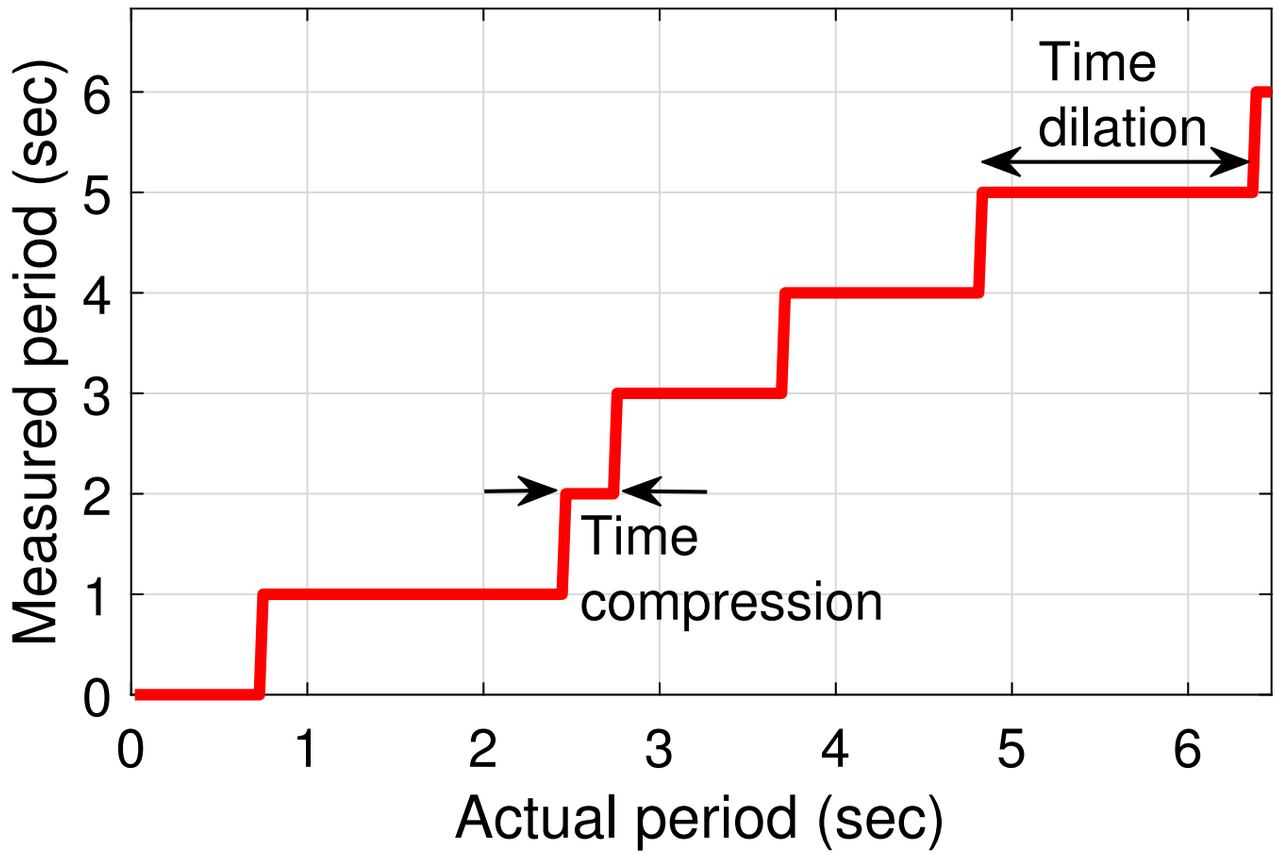
Attack on SGX “Trusted” Time

Result:

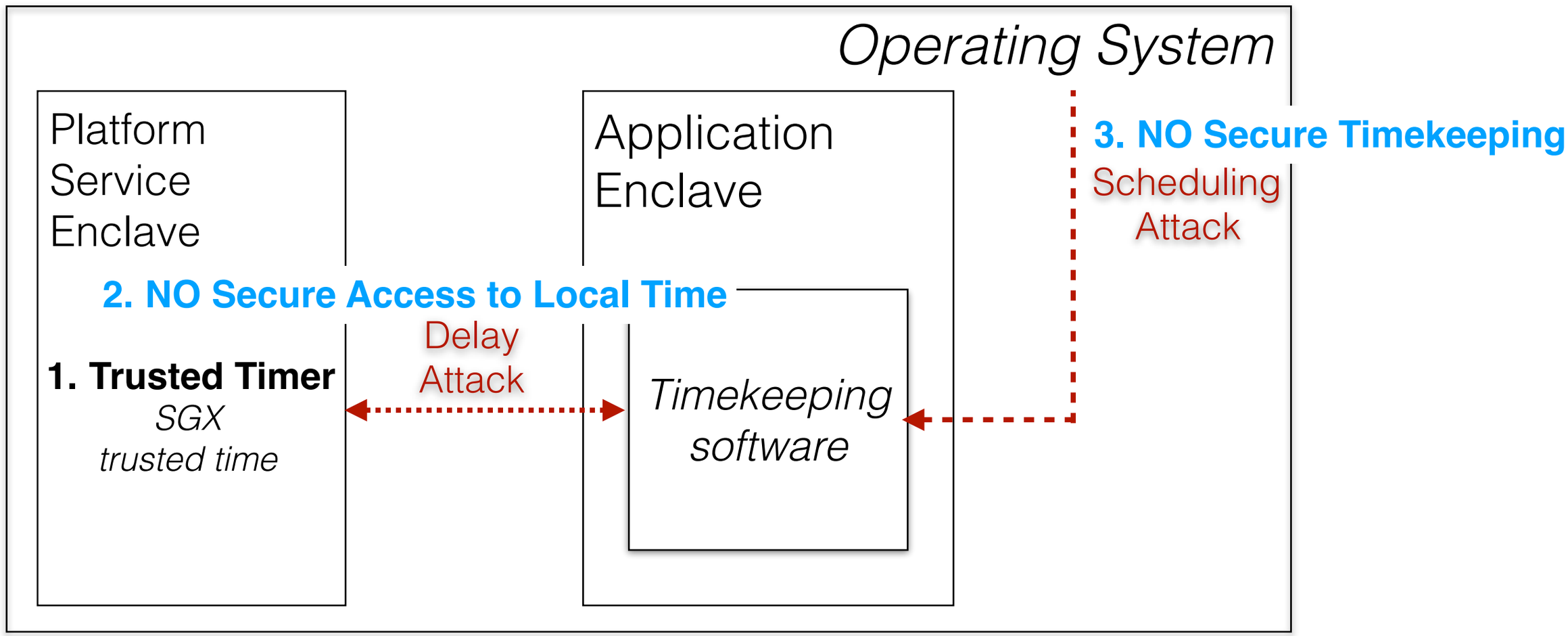


Attack on SGX “Trusted” Time

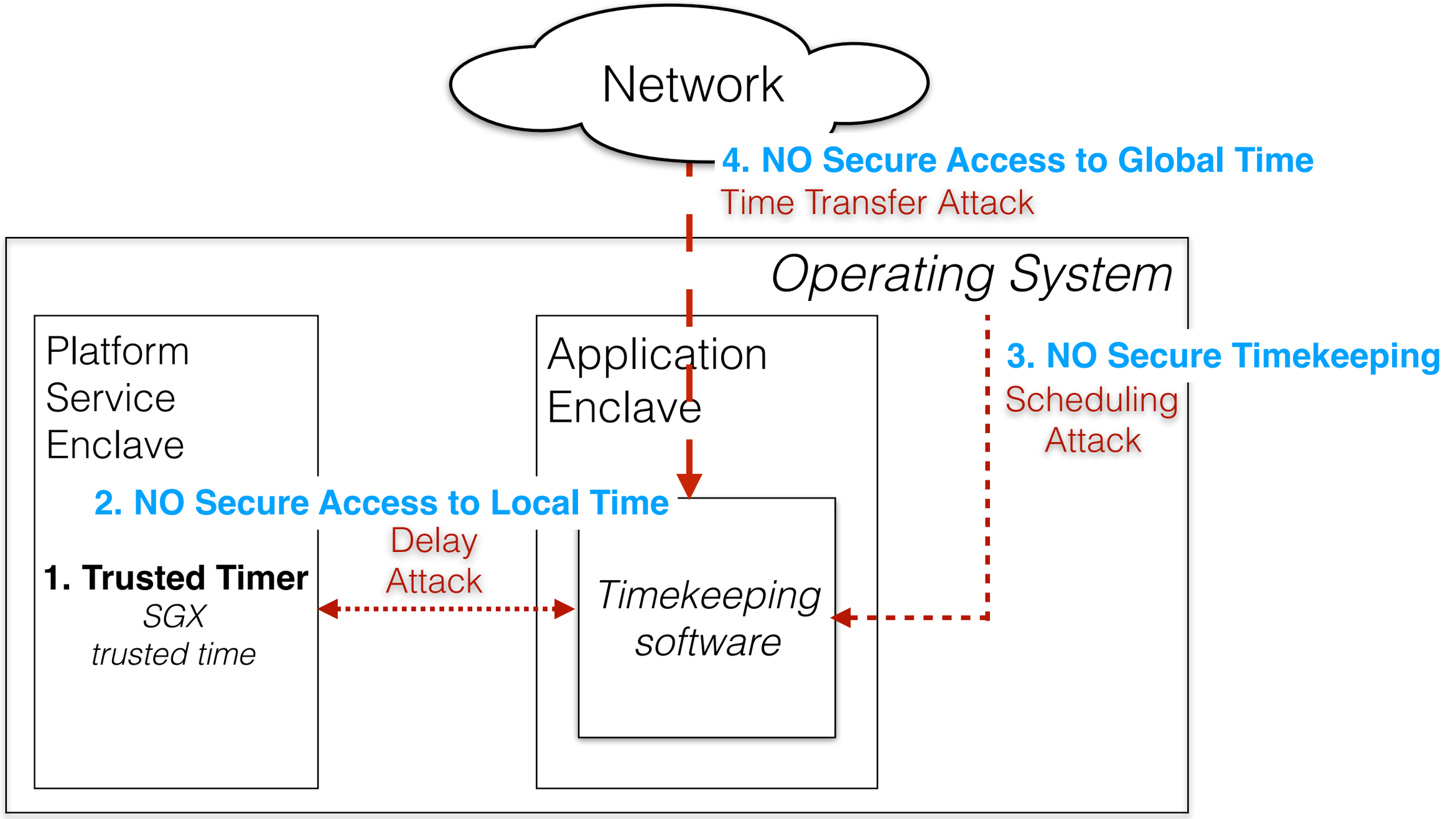
Result:



Attack on SGX “Trusted” Time



Attack on Network Packets



TimeSeal: A Secure Time Architecture

1. Build High Resolution clock
2. Overcome OS Scheduling attacks
3. Compensate for Delay attacks

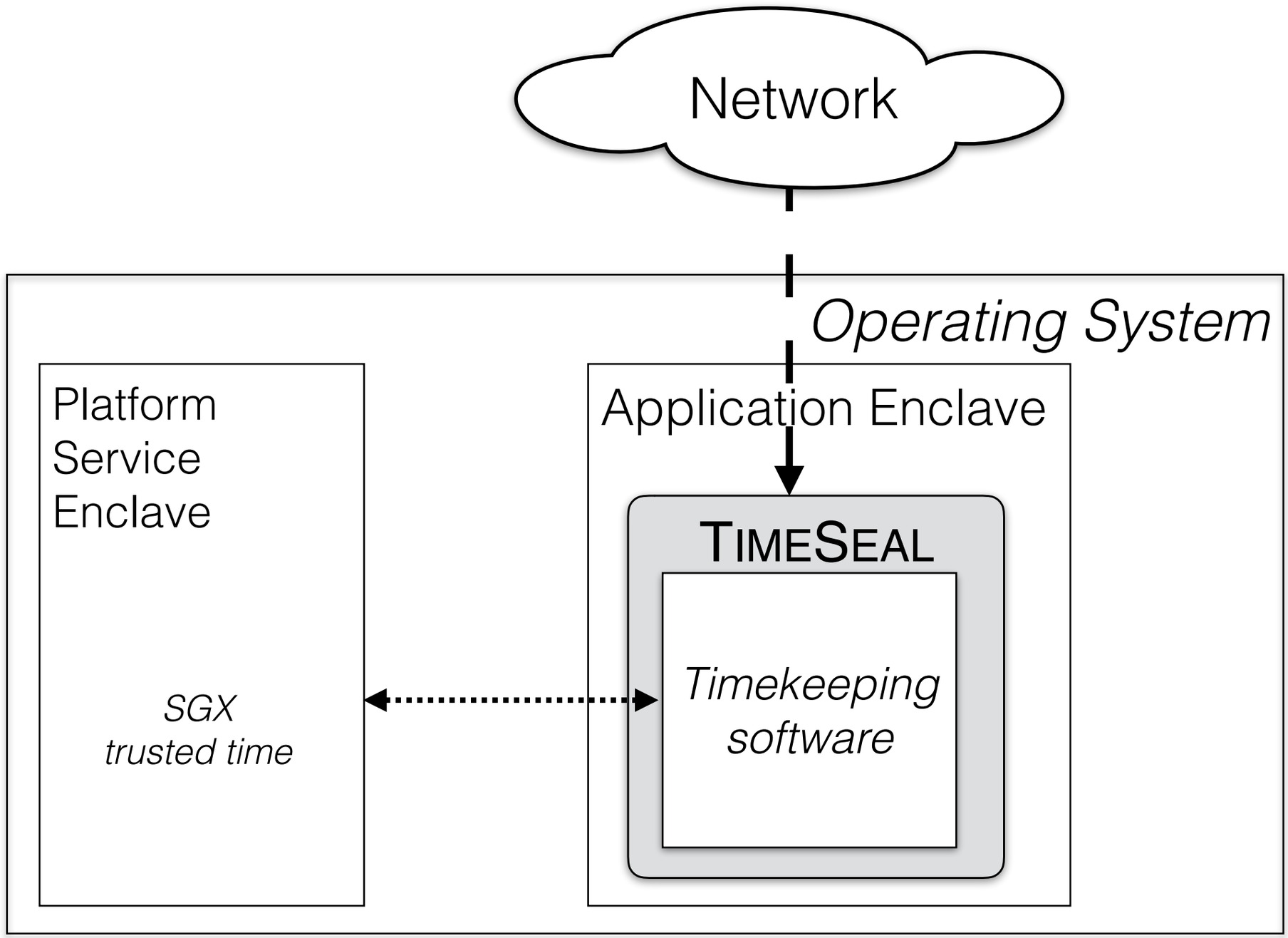
} Secure Local Clock

4. Mitigate Time Transfer attacks

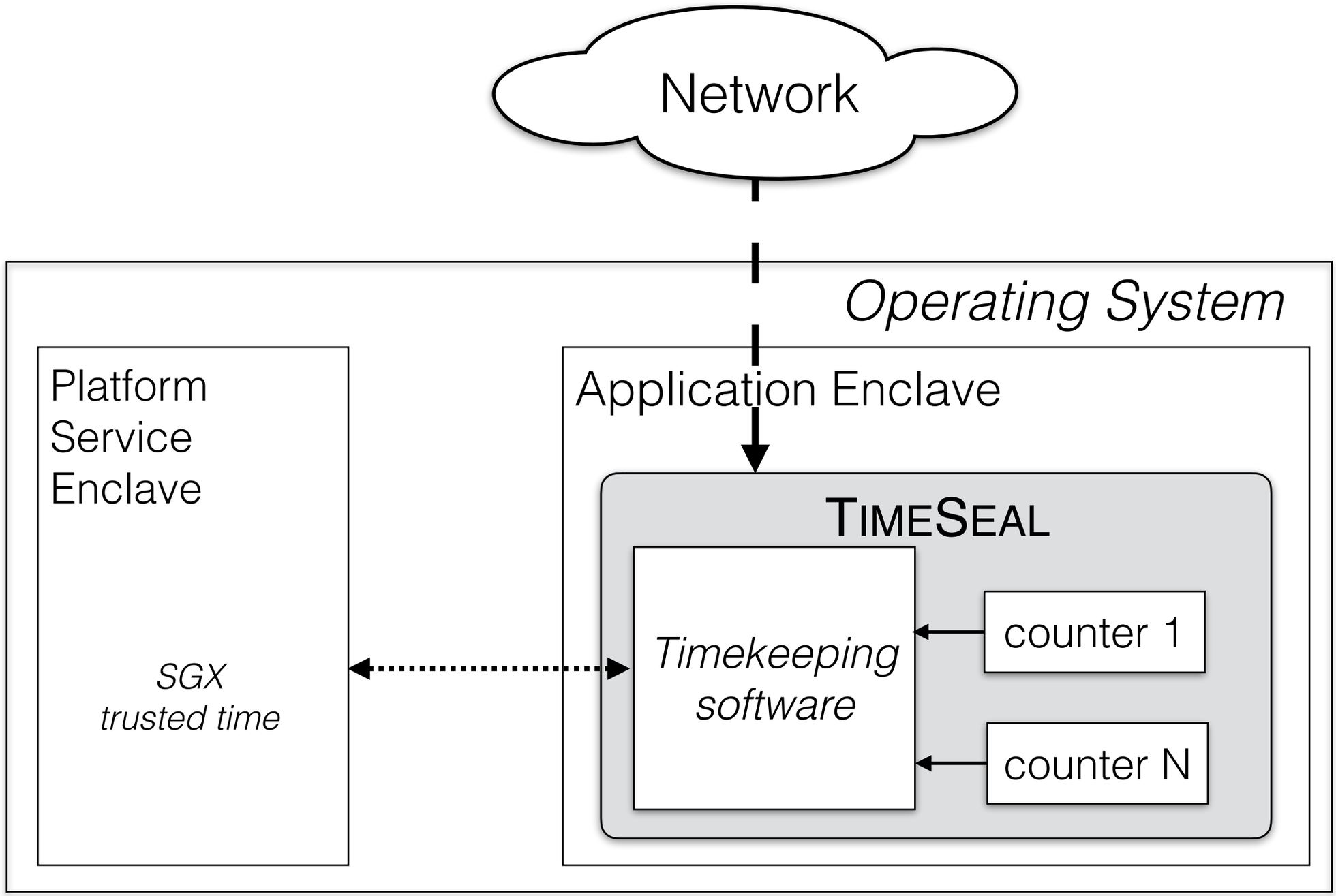
} Secure Global Clock

Prototyped TimeSeal on Intel SGX

TimeSeal: A Secure Time Architecture

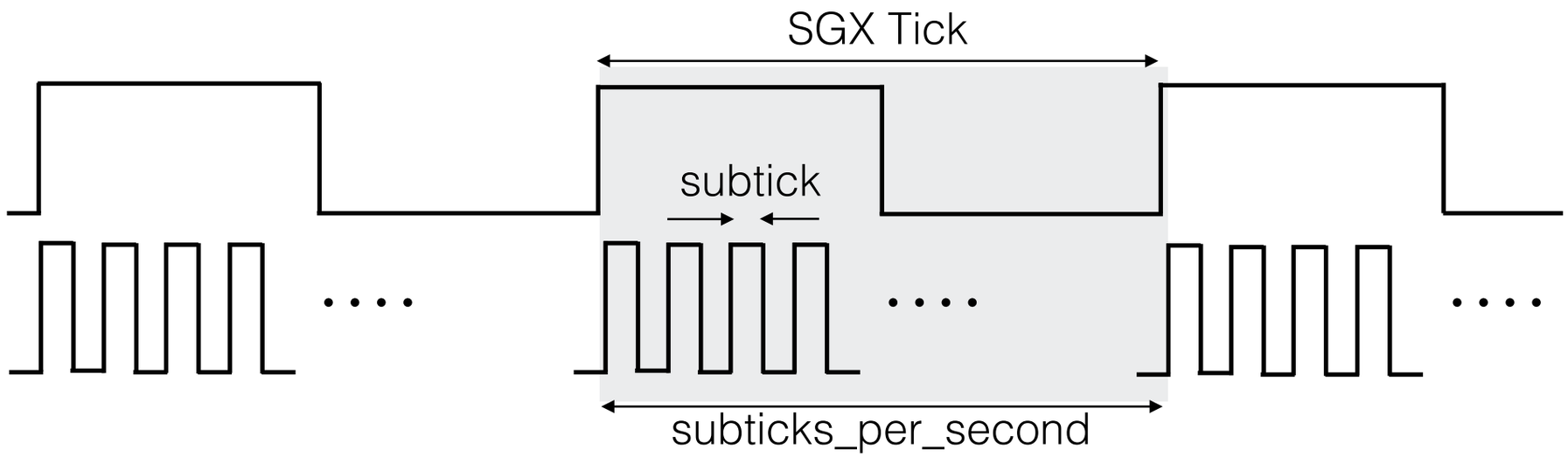


1. High Resolution SGX Clock

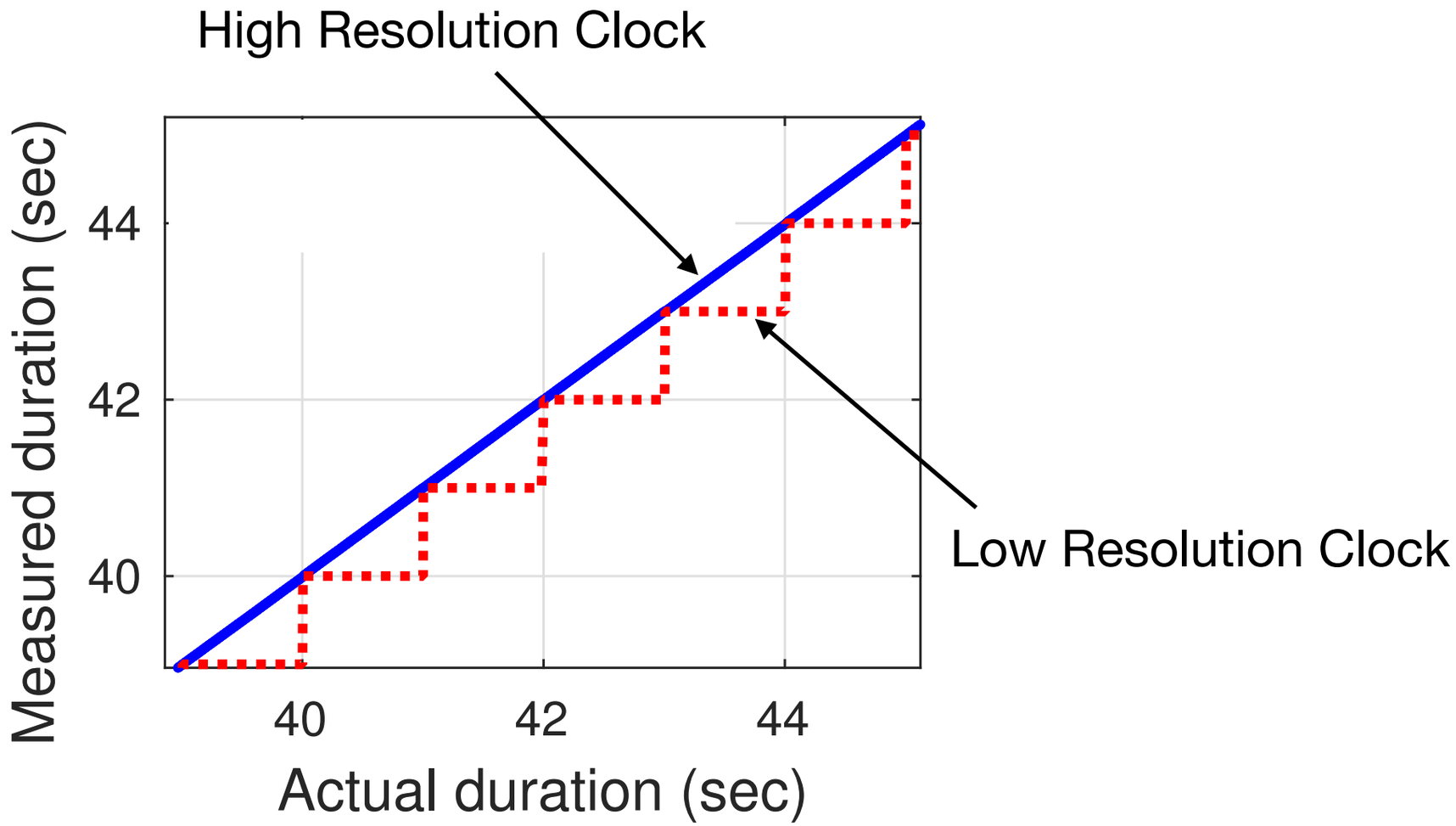


1. High Resolution SGX Clock

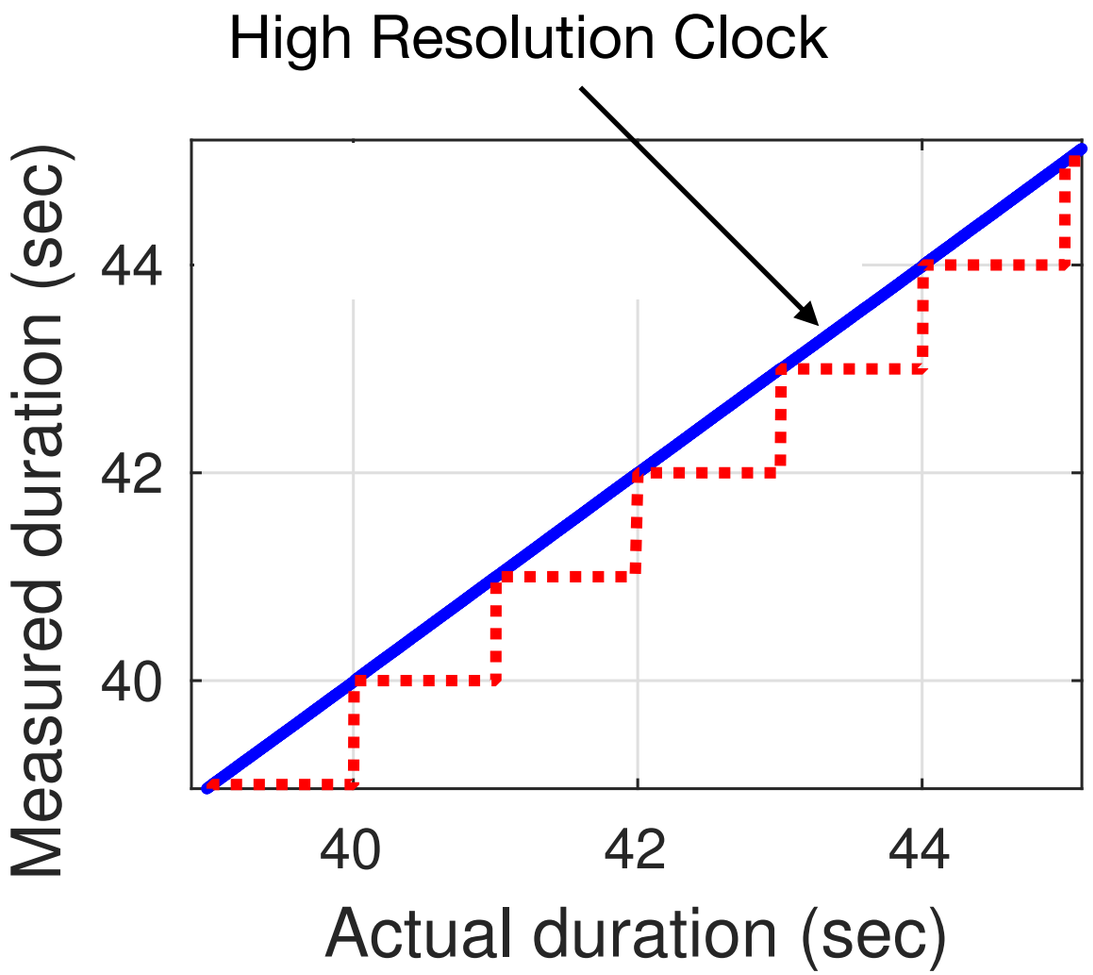
$$t_{local} = SGXticks + \frac{subticks}{MA(subticks\_per\_second)}$$

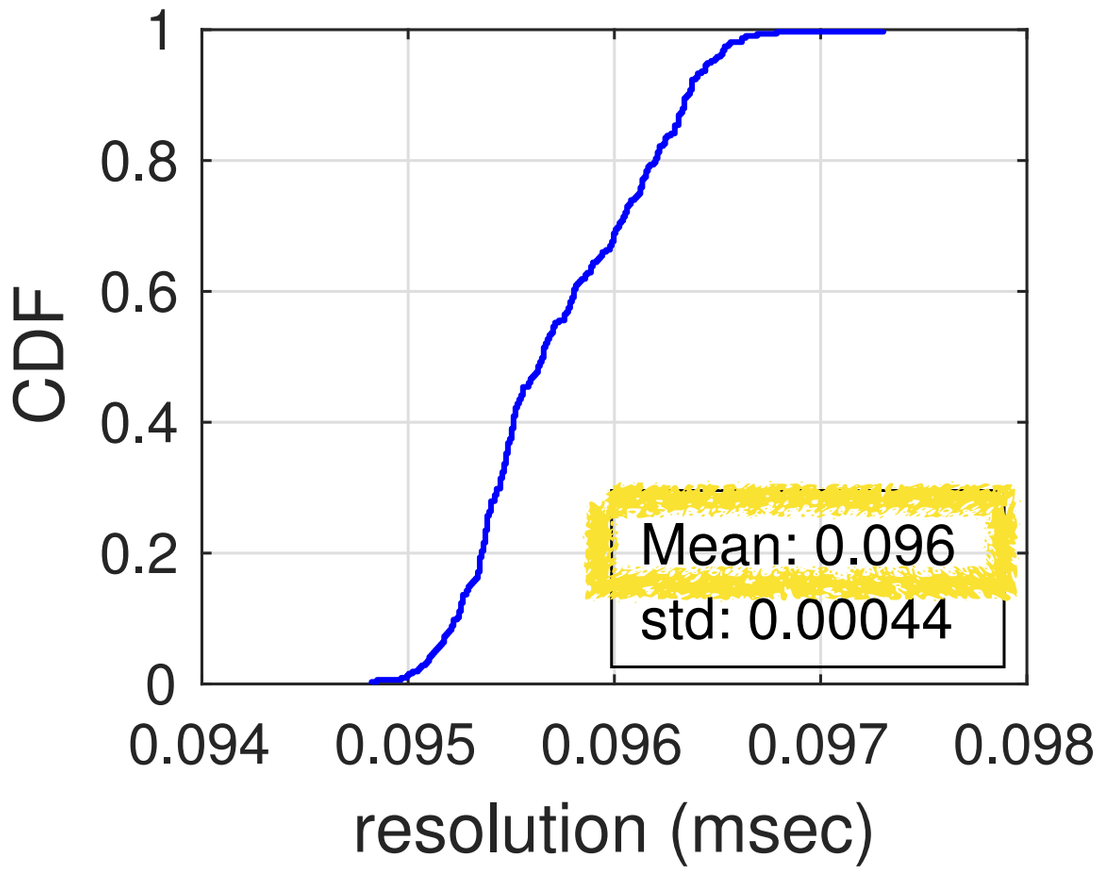


1. High Resolution SGX Clock



1. High Resolution SGX Clock

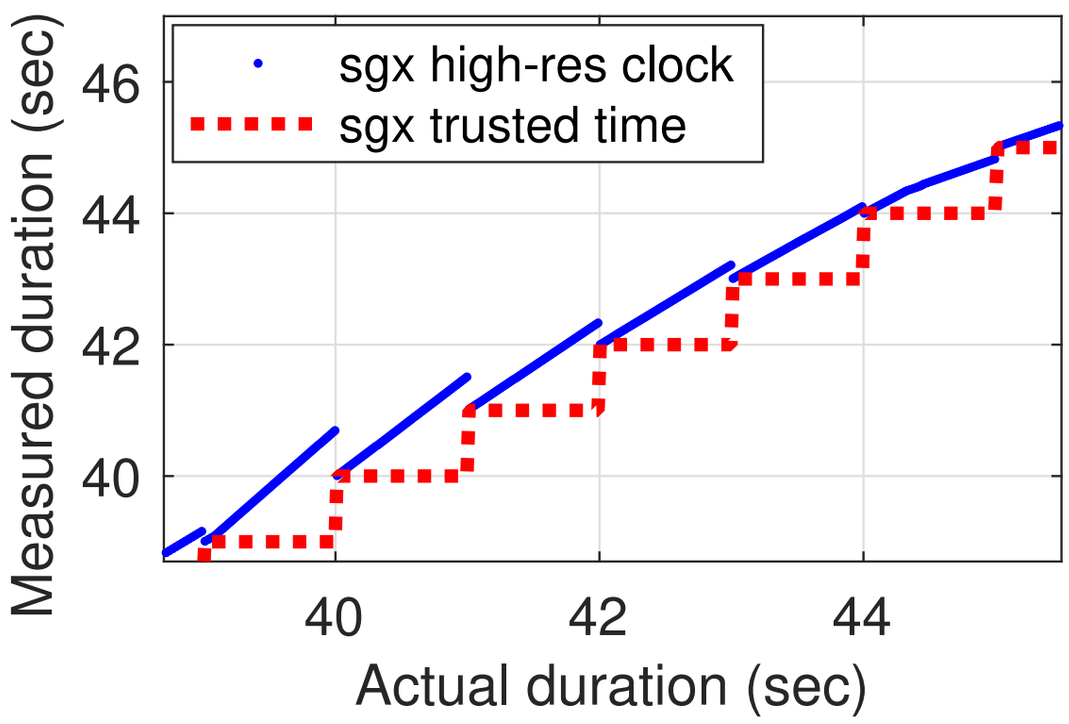


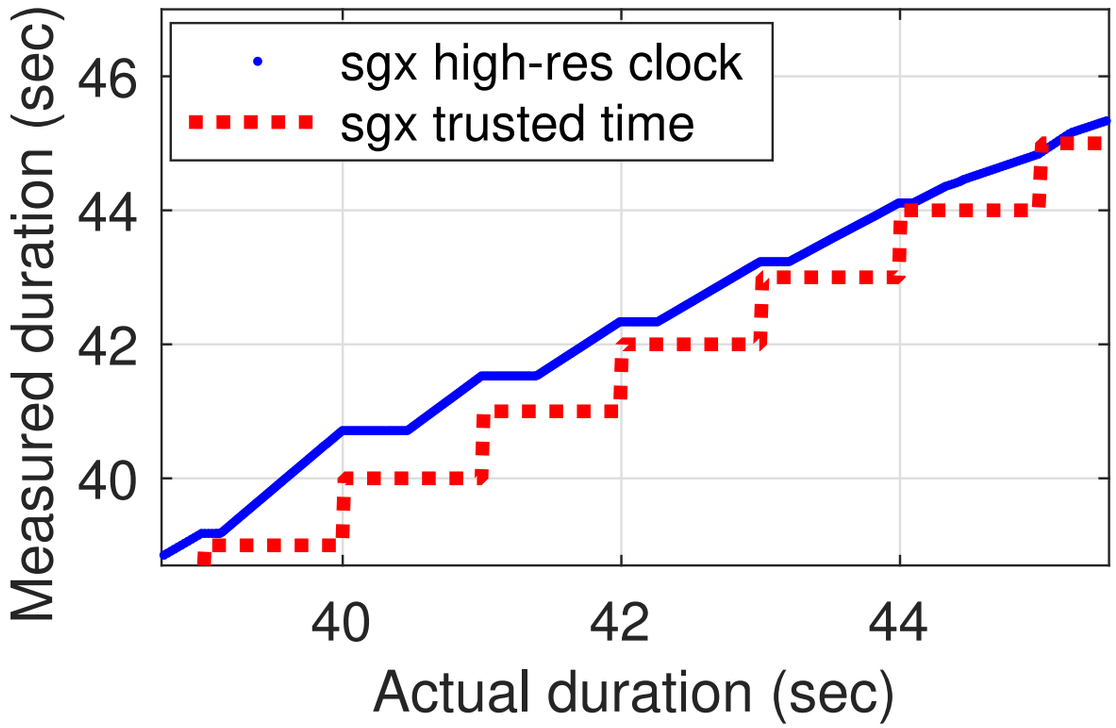


1. High Resolution SGX Clock

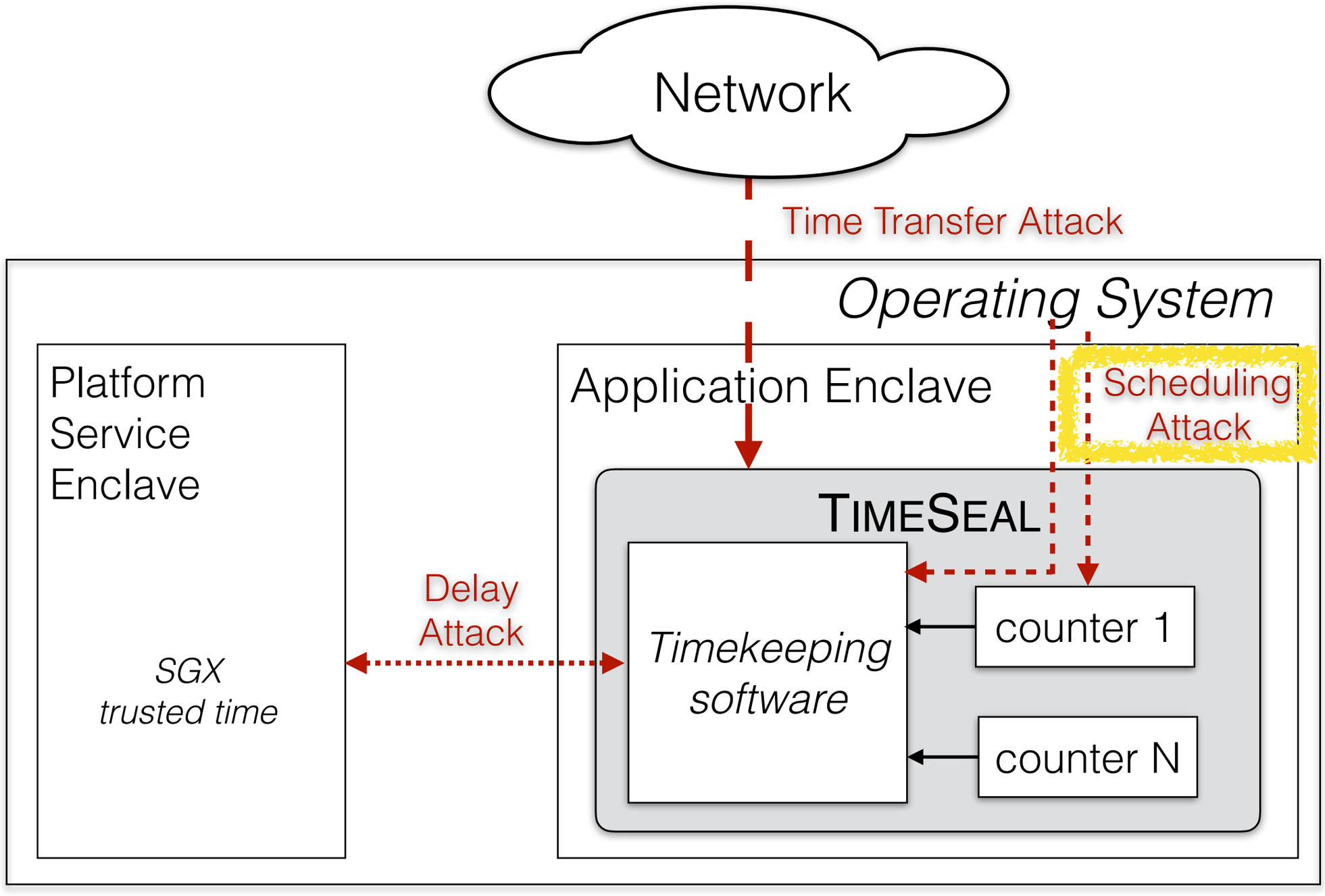
High System Load

$$t_{local} = t_{prev\_local} + \frac{subticks}{MA(subticks\_per\_second)} - slew(t_{local\_at\_SGXtick} - SGXtick)$$





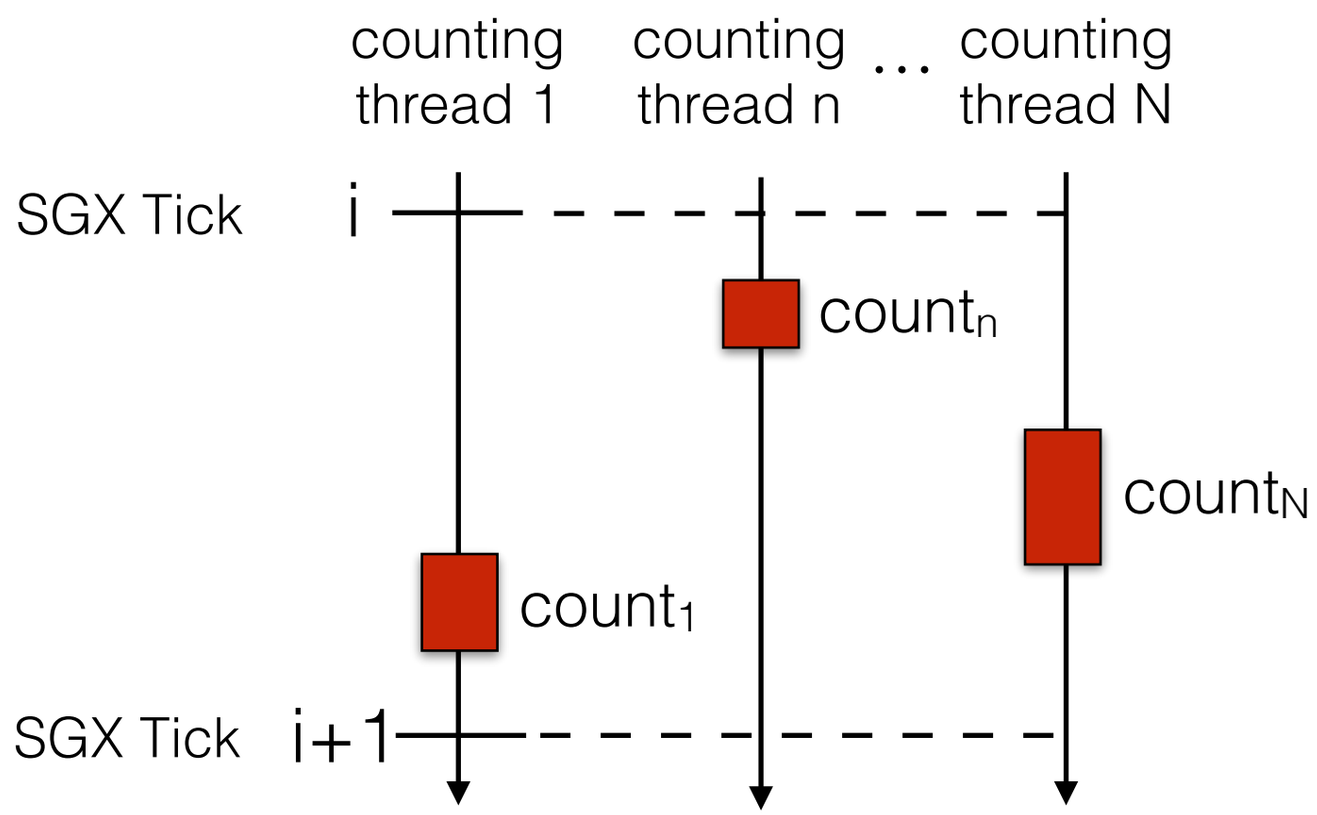
TimeSeal Overcomes OS Scheduling Attacks



Overcome OS Scheduling Attacks

Intuition:

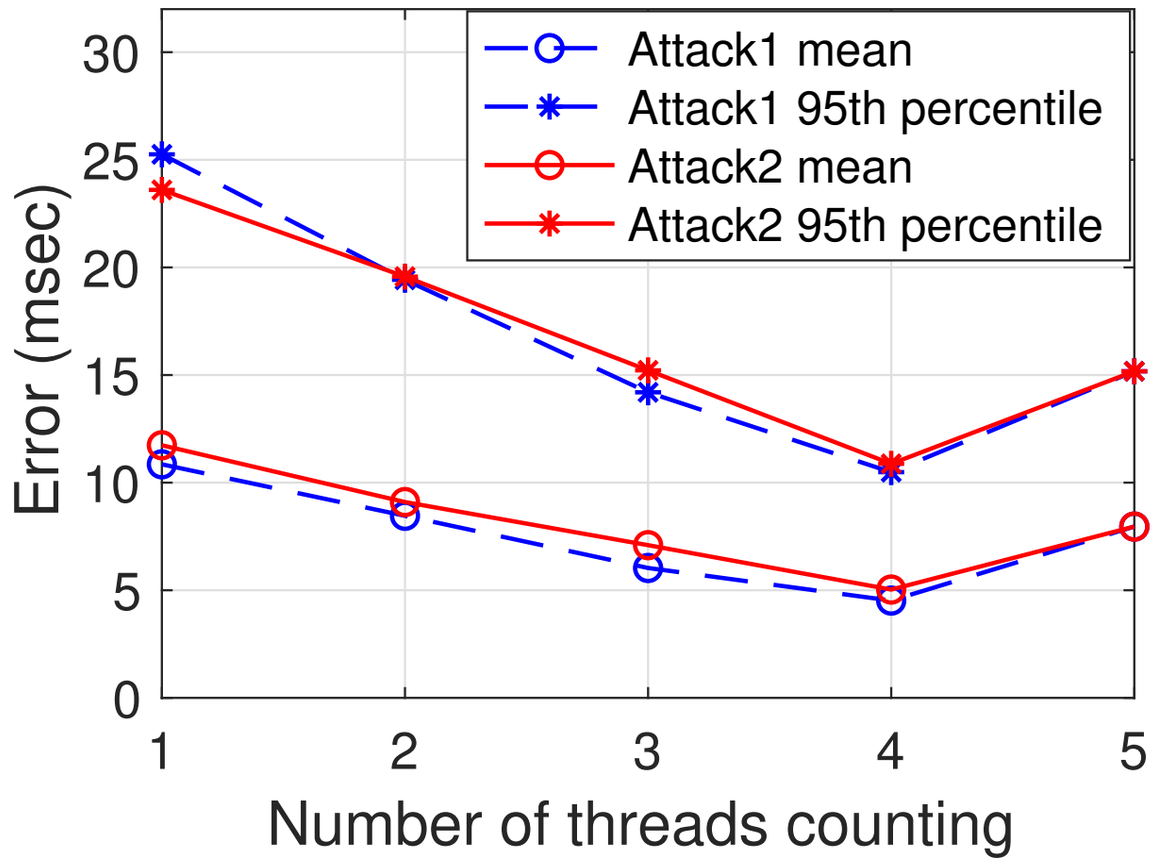
Randomize the *counting thread order (n)* and *count interval ($count_n$)*



$$P(resolution\ degradation) = f(n, N, count_n)$$

Secure Timekeeping Software

Mitigate Scheduling Attack

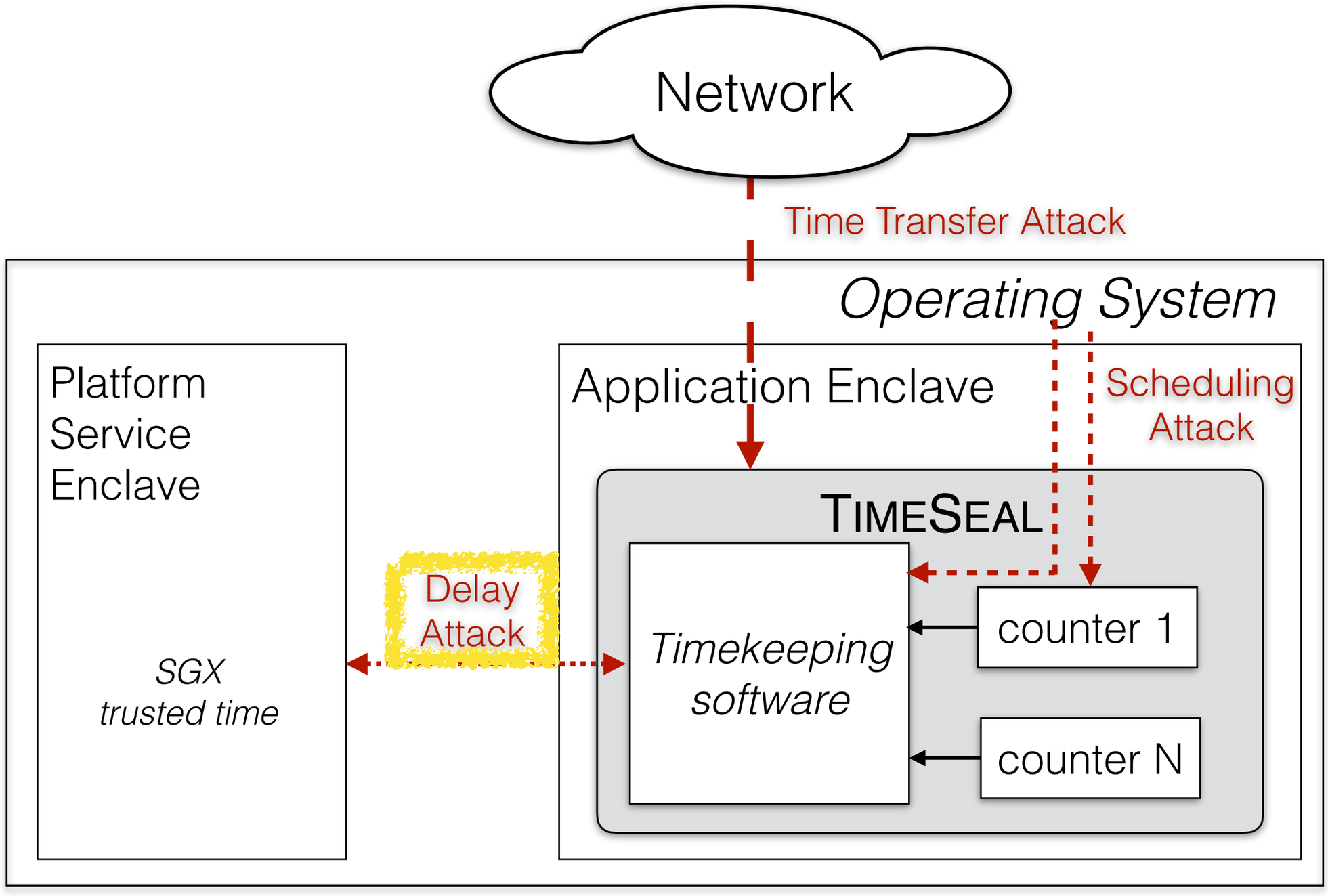


Secure Timekeeping Software

Mitigate Scheduling Attack

	Attack 1				**Attack 2**				**Attack 3**	
	Single Thread		N-1 Threads		Single Thread		N-1 Threads		50% c_d	
	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
Policy A	0.5|| 3	7|| 7	0.5|| 3	9|| 8.4	0.5|| 1.8	7|| 7	0.5|| 2	9|| 8.5	0.6|| 8	25|| 24
Policy B	0.002|| 3	6.6|| 5.2	0.068|| 3	9.6|| 8.5	0.076|| 1.5	6|| 5	0.05|| 0.6	10|| 7.4	0.08|| 9.4	22|| 24
Policy C	8|| 2.7	300|| 200	0.3|| 0.6	1500|| 1600	1|| 11	81|| 60	1|| 25	100|| 99	0.02|| 5	31|| 26

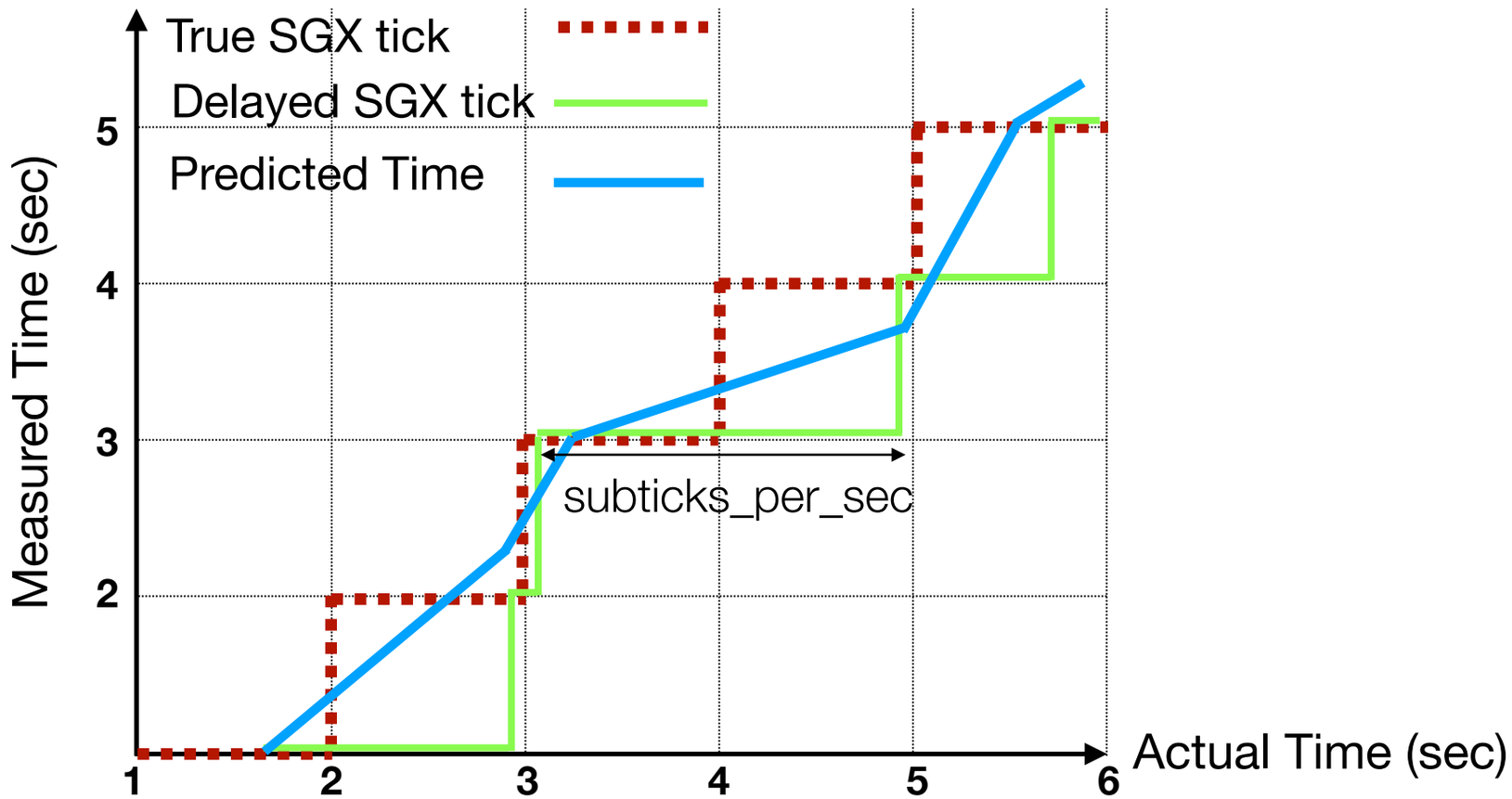
TimeSeal Compensates for Delay Attacks



Compensate for Delay Attacks

Intuition: Leverage domain specific knowledge

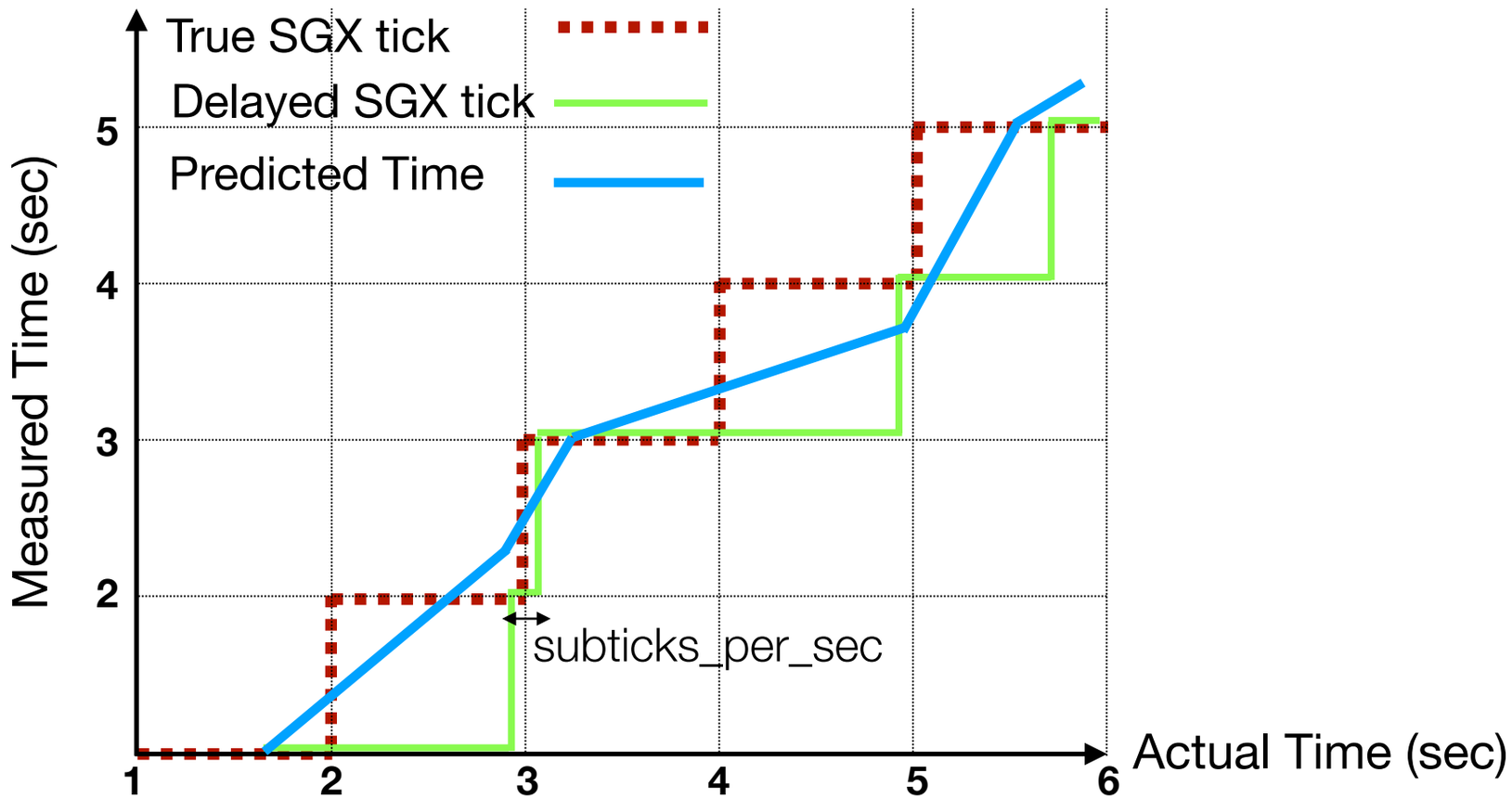
$$t_{local} = t_{prev\_local} + \frac{subticks}{MA(subticks\_per\_second)}$$



Compensate for Delay Attacks

Intuition: Leverage domain specific knowledge

$$t_{local} = t_{prev\_local} + \frac{subticks}{MA(subticks\_per\_second)}$$

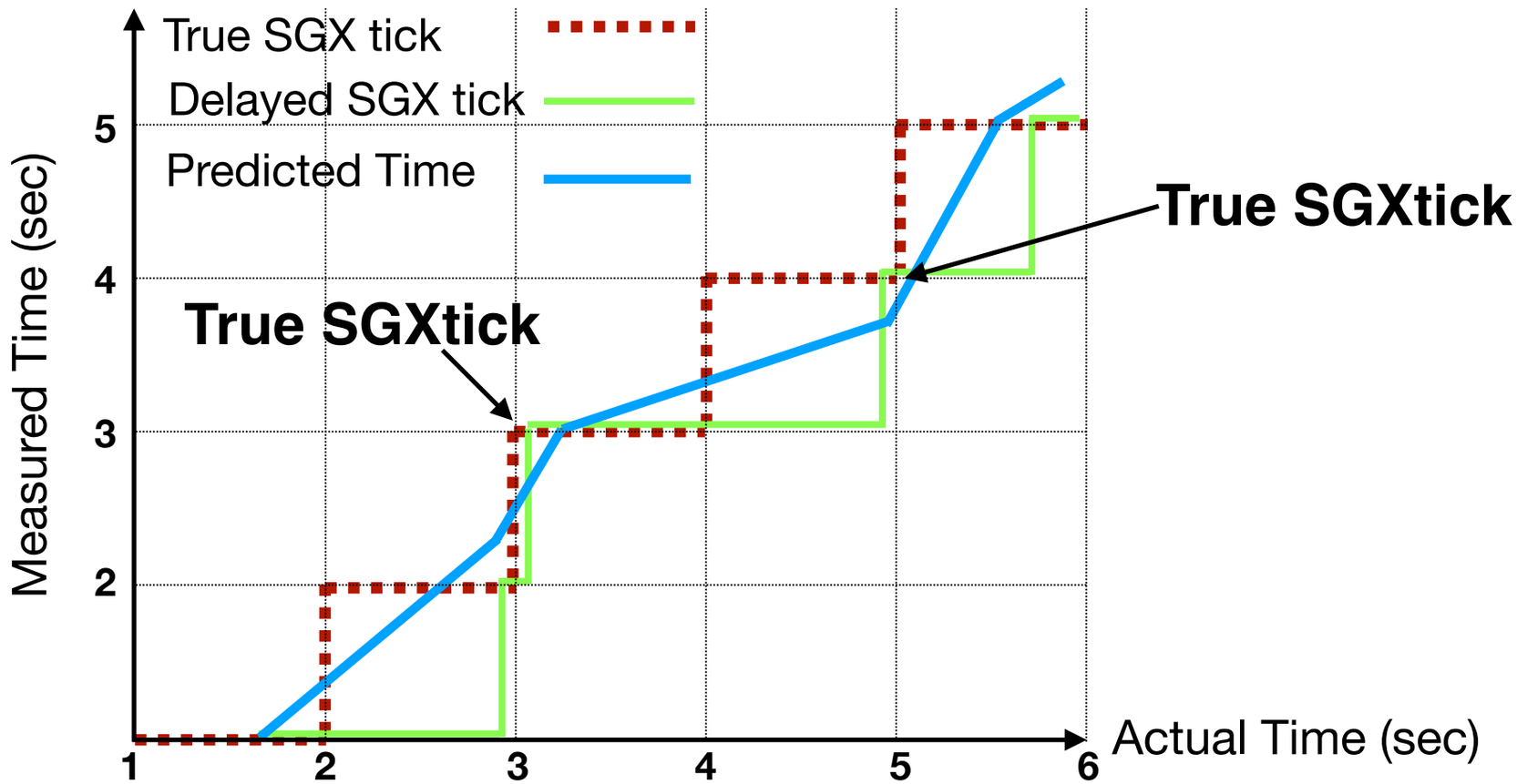


Compensate for Delay Attacks

Intuition: Leverage domain specific knowledge

Bound *True SGX ticks to compensate for error*

$$t_{local} = t_{prev\_local} + \frac{subticks}{MA(subticks\_per\_second)}$$

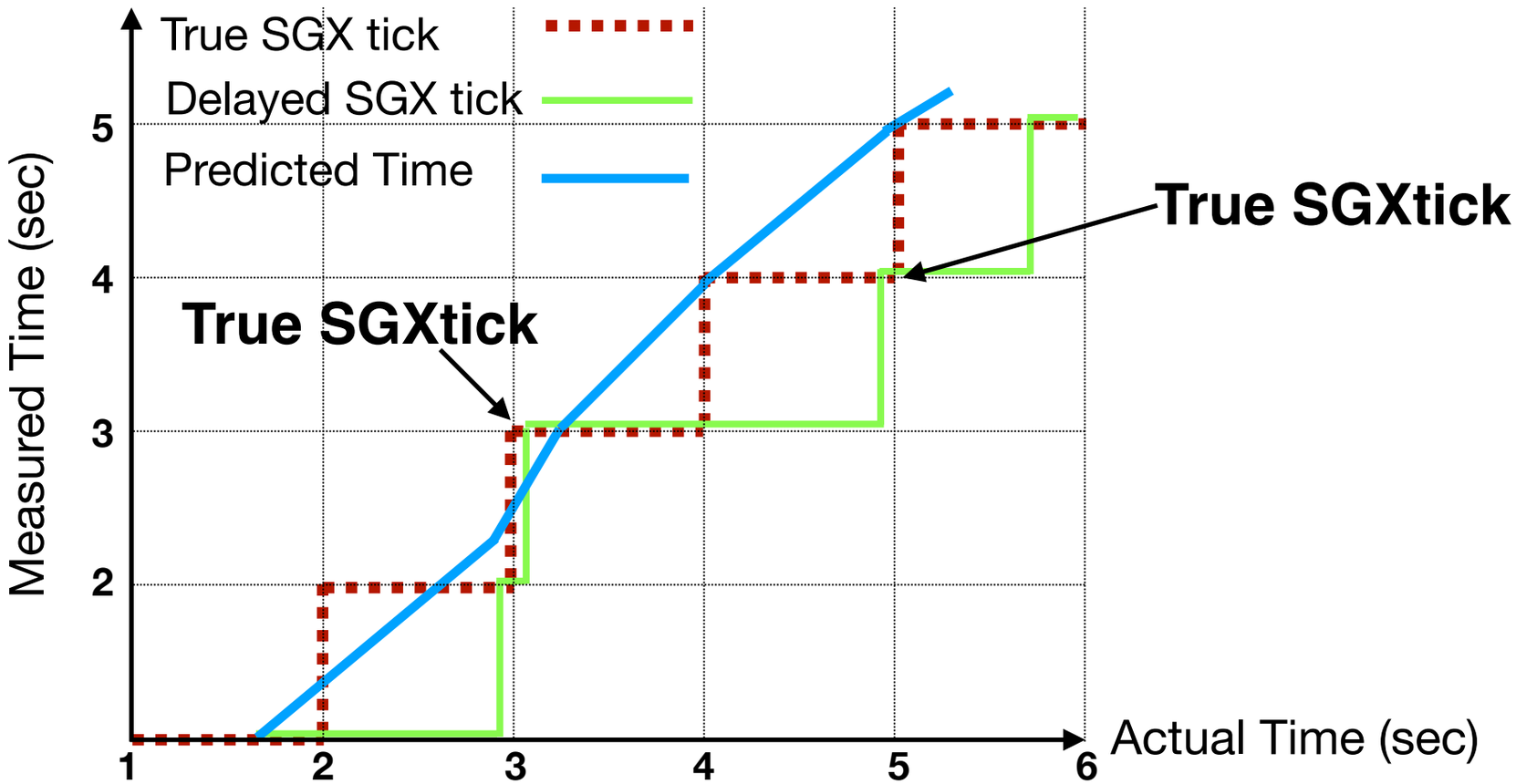


Compensate for Delay Attacks

Intuition: Leverage domain specific knowledge

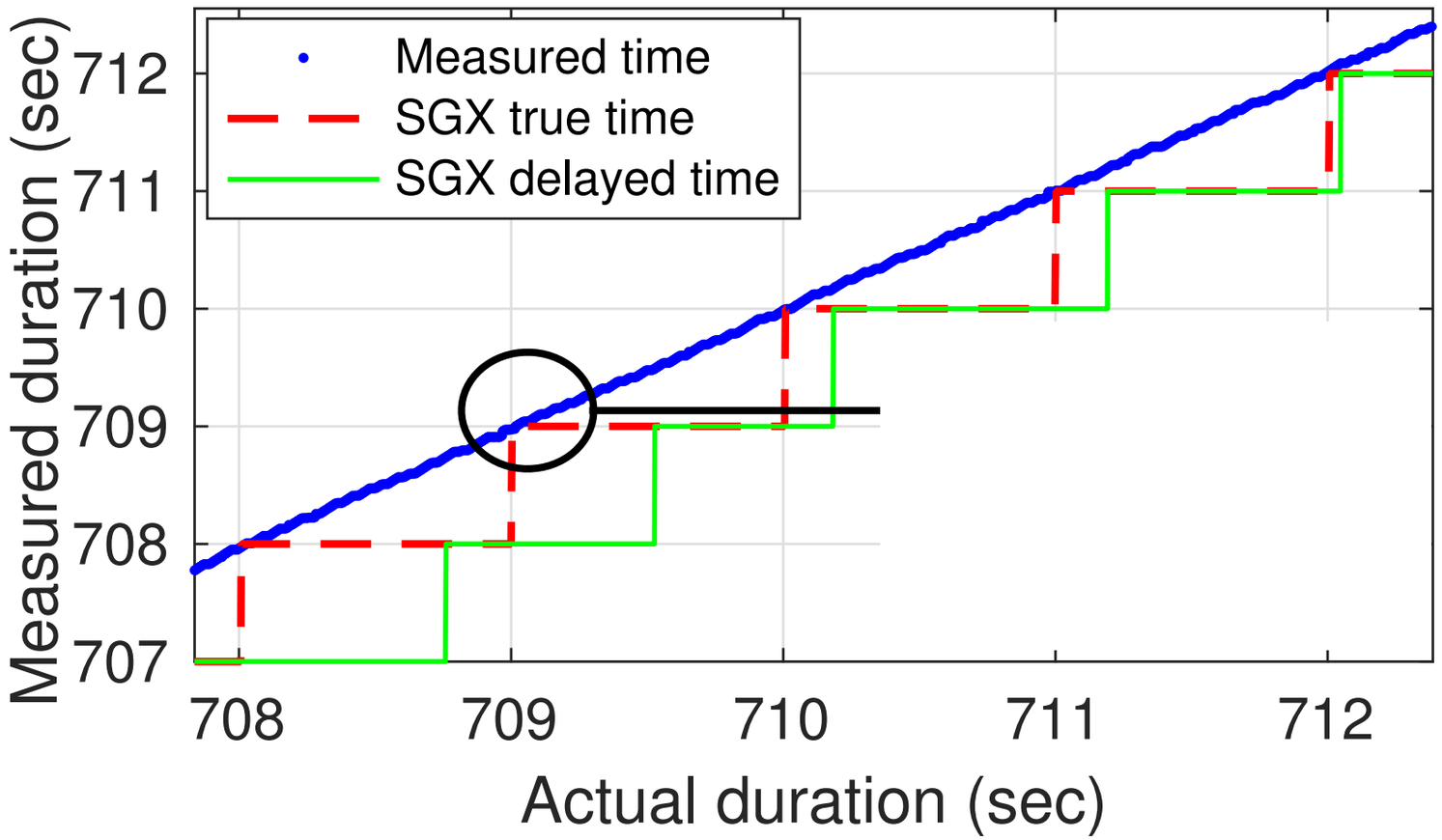
Bound *True SGX ticks to compensate for error*

$$t_{local} = t_{prev\_local} + \frac{subticks}{MA(subticks\_per\_second)} - slew(t_{local\_at\_SGXtick} - SGXtick)$$



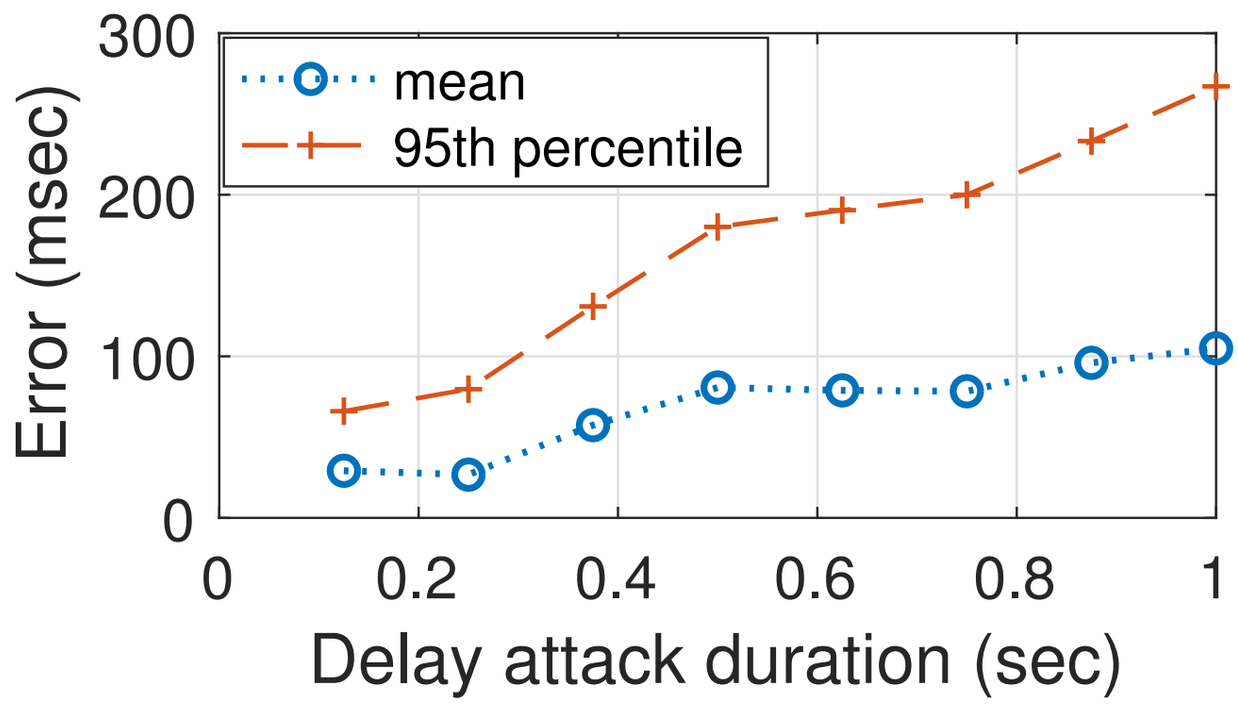
TimeSeal Evaluation

Mitigate Delay+Scheduling Attacks



TimeSeal Evaluation

Mitigate Delay+Scheduling Attacks



TimeSeal: A Secure Time Architecture