Device Attestation and Malware analysis

PI: Prof. Brejesh Lall **CO-PI:** Dr. Sk Subidh Ali **PS:** Debanka Giri

Problem Statement: Design a mechanism to remotely verify a potentially untrusted device has been compromised or not i.e., check software integrity as well as legitimate operation.

Tools Used: (i)Raspberry pi 4 loaded with Secure memory like TPM
(ii)LDR, Fan, Alarm Buzzer, LED etc.
(iii) Python

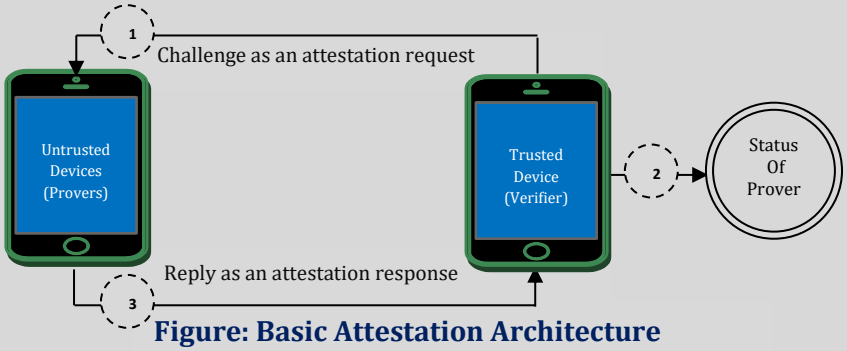
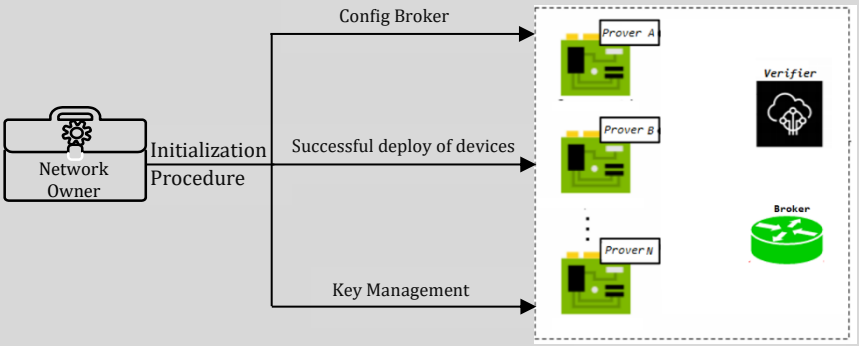


Figure: Basic Attestation Architecture

Scope: The device attestation model developed for the Swarm Network employs distributed architecture as a part of **Indigenous 5G test-bed**. Our proposed protocol verifies trustworthiness of the IoT devices as well as the legitimate operations among them. The protocol can detect the adversaries and attacks like Software adversary, Mobile adversary and replay attack. The Denial of service (Dos) is out of our current scope.

Demonstration Details: To demonstrate the attestation protocol functionality we have integrated different sensors with Raspberry PI and acts as an IoT Device. We have configured RPi in such a way such that after booting RPi runs integrated sensor automatically. Then we disconnect any one of the RPi and manipulate the sensor program and restart again and rejoin the network. Afterward verifier detect the compromised service.



Entities in Swarm Network

Protocol Description: Our protocol consists of three main phases
(i) **Deployment Page:** This offline phase executed by network owner to ensure secure and successful deployment of each devices, key distribution as well as set up verifier with desired configuration.
(ii)**Attestation Phase:** At first **Verifier** initiates the attestation protocol by sending an attestation challenge to Prover. Upon receiving attestation request prover execute concern operations such as Global Hash and Local Hash calculation etc. Afterward, Prover forward the attestation result to verifier or other provers based on service dependency.
(iii)**Verification Phase:** The verification starts when the Verifier retrieves the attestation result. Afterward verifier analyze the attestation result and verifies the hash values of each services of provers.

Protocol steps

Verifier

$(R) = (0,1)^n$
$Ch = \sigma_{ver}$=sig (Skey_ver; P||R)

Publisher (p) (Prover)

If(verify_sig(Pkey_ver; ; P||R, σ_{ver}) then
Begin:
Calculate the values : $Timestamp_p$, ServID, GHV_{PREV}, $Input_P$, $Output_P$
Attest()
LHV_P = checksum(p);
$Timestamp_P[p] = Timestamp_P[p] + 1$
τ = ServID|| $Timestamp_P[p]$|| LHV_P||
$Output_P$ ||$Input_P$|| GHV_P
GHV_P =Enc(Pkey_ver, τ)
Publish()
msg_p = $Output_P$ || GHV_P|| $Timestamp_P[p]$
σ_p=sig (Skey_p; P||R)
Else:
Reject Ch
End

Subscriber (S) (Prover)

If(verify_sig(Pkey_p; σ_p, msg_p) then
Begin:
Calculate the values : $Timestamp_s$, ServID, GHV_{PREV}, $Input_s$, $Output_s$
Attest()
LHV_s = checksum(s);
$Timestamp_s[s] = Timestamp_s[s] + 1$
τ = ServID|| $Timestamp_s[s]$|| LHV_s||
$Output_P$ ||$Input_P$|| GHV_P
GHV_s =Enc(Pkey_ver, τ)
Else:
Reject σ_p, msg_p
End

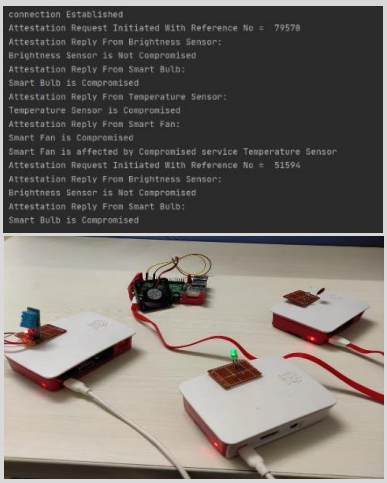


Figure: Attestation Demonstration

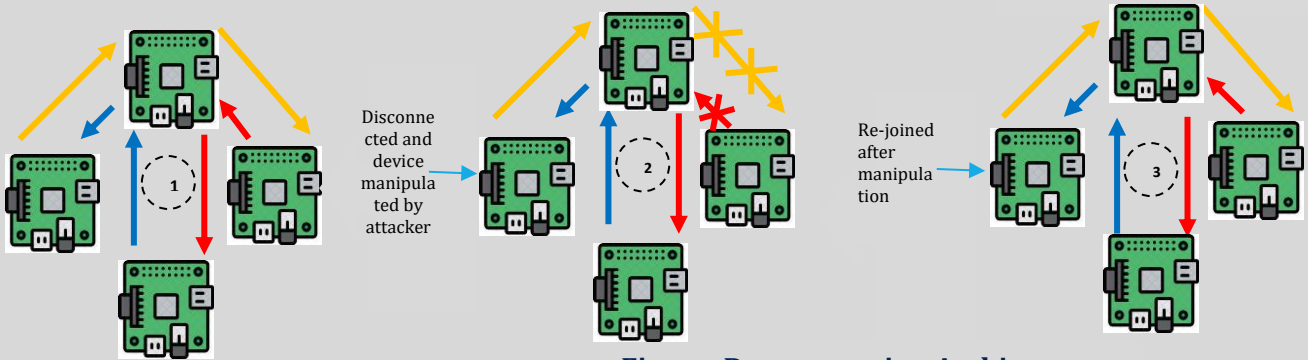


Figure: Demonstration Architecture