Cryptography and Tera TVMM

Public Key Cryptography
-Symmetric key cryptography is when two people use same key to encrypt and decrypt a message.
-To break that, each person has a public key, which can be given to anyone, and a private key, which is kept secret.

Alice
\[ P_A - \text{Public Key} \]
\[ S_A - \text{private Key} \]

To encrypt a message to Alice
\[ C = E(P_A, M) \]

To read message
\[ M = D(S_A, C) \]

RSA
To generate public / private key Alice does
1. Pick primes p, q
2. \[ N = p \times q \], \[ P(N) = (p-1)(q-1) \]
3. Find e and d s.t. \( (e \times d) = 1 \mod P(N) \)
4. \[ P_A = (e, N) \]
   \[ S_A = (d, N) \]

RSA Encryption and Decryption
\[ C = E((e, N), M) = M^e \mod N \]
\[ M = D((d, N), C) = C^d \mod N \]
\[ = M^{ed} \mod N \]
(Fermat’s Little Theorem) = M

\[ N = 35 \], \[ P = 7 \], \[ P(N) = 24 \], \[ q = 5 \], \[ e = 7 \], \[ d = 7 \] in real systems e and d different

\[ P_A = (7, 35) \]
\[ S_A = (7, 35) \]

\[ E(P_A, 2) = 2^7 \mod 35 = 23 \]
Public key signatures
Each user has private signing key and a public verification key, $S_A$ and $P_A$ respectively.

$S = \text{Sig}(S_A, M)$

$\text{Verify}(P_A, M, S) = \text{Valid or Invalid}$

To generate public / private key Alice does

1. Pick primes $p$, $q$
2. $N = p \cdot q$, $P(N) = (p - 1)(q - 1)$
3. Find $e$ and $d$ s.t. $(e \cdot d) = 1 \mod P(N)$
4. $P_A = (e, N)$
   $S_A = (d, N)$

$\text{Sig}((d, N), M) = M^d \mod N$

$\text{Verify}((e, N), M, S) = \text{Valid iff } S^e = M \mod N$

Signing Long Messages
- Use hash function

$S = \text{Sig}(S_A, h(m))$

$h : \{0,1\}^* \text{ binary string of any length} \rightarrow \{0,1\}^n$

Strong Collision Resistance
$h$ is S.C.R. if it is hard to find $x \neq y$ such that $h(x) = h(y)$

Certificates
To verify Alice’s signature, Bob needs to know $P_A$. Suppose Alice and Bob trust Charlie

$C_A^C = (\text{Sig}(S_C, \text{ “Alice public key in } P_A\text{”}))$
**Terra, TVMM**
-The owner of the computer is malicious
-Need “root-secure” system
-Applications
  -Network Games
  -Movie/Music Players (aka Digital rights Management (DRM))
  -Reverse Engineering

**Trusted Virtual Machine Monitor**

<table>
<thead>
<tr>
<th>VM 1</th>
<th>VM 2</th>
<th>VM n</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apps</td>
<td>Apps</td>
<td>…</td>
</tr>
<tr>
<td>OS</td>
<td>OS</td>
<td></td>
</tr>
</tbody>
</table>

Virtual Machine Monitor

Hardware

- VMM exposes hardware interface to guest VMs
- Strong isolation between VMs
  - Memory
  - CPU
  - Disk
  - Network, etc

**Remote Attestation**
Goal: Prove to a remote party that I am running a certain set of software: App
If attacker can circumvent any part of those software layers then it can fool other system in believing that it is running a certain set of software.