A Security Infrastructure for Trusted Devices

Mahalingam Ramkumar
Mississippi State University, MS
Nasir Memon
Polytechnic University, Brooklyn, NY

January 31, 2005

Emerging Models of Trust

- Paradigm shift in the model of trust in emerging applications
- Conventional applications - Client-server applications
  - End users are trusted
  - Trusted not to reveal passwords, private keys
  - In theory, compromise of user A's secrets should not affect *other* users
- Pervasive / ubiquitous computing, ad hoc networks, DRM
  - devices need to be trusted
  - to behave in a "responsible fashion"
  - not the "owners" or "operators."
- How do we trust devices?
- More appropriately, how do *devices trust each other*?

Trusted Devices

- Devices “play by the rules”
- Compliance to established rules.
- How?
  - Trusted devices provided with secrets
  - Secrets serve as a “hook” for compliance
  - Verify compliance *before* providing secrets
  - Verification of (possession of) secrets = verification of compliance
- Mechanism to distribute and establish possession of secrets - key distribution scheme (KDS)

Tamper Resistance and Read Proofness

- Even "owners" of the devices should not have access to the secrets
- Devices are *trusted not to reveal their secrets*!
- Both tamper resistance and read-proofness are *mandatory*
- Tamper resistance - guarantees that components that guarantee compliance cannot be modified *after* a device is provided with secrets
- Read proofness - guarantees that secrets from a compliant device cannot be *transferred* to a non-compliant device

Renewability

- Technology for tamper-resistance is expected to improve (necessity is the mother of invention!)
- Yet perfect tamper resistance / read proofing may never be achievable
- Need to *renew* secrets periodically

Safe Renewal of Secrets

- Secrets originally assigned by the manufacturer
- Take the device back to the manufacturer every time for renewal? - not practical
- Renewal has to occur over *open channels* (Internet?)
- Devices will authenticate themselves using old secrets to receive new secrets
- If old secrets in a device have been compromised, what prevents an attacker from getting new secrets?
- Need an additional secret that cannot be compromised by tampering.
- No, password is not sufficient.

Circuit-Delay Based Authentication

- B. Gassend, D. Clarke, M. van Dijk, S. Devadas, "Delay-based Circuit Authentication and Applications," Proceedings of the 2003 ACM symposium on Applied Computing, Melbourne, Florida, pp 294 – 301, 2003.
- Uncontrollable delays unique to each chip can serve as a signature
- Not exposable by tampering
- Sensitive to environmental variations - could be compensated
- Possibly weak secret

Safe Renewal

- Assumptions
  1. The existence of a weak secret which **cannot** be exposed by tampering.
  2. The only way to obtain secrets from a device A is by *tampering* with the device A.
  3. Devices that are tampered with are rendered *unusable* in the future.
- Safe renewal is feasible!
- The key renewal process (protocol) can de set up such that each brute force attempt would need TA's involvement!

KDS Requirements

- Extremely large scale (billions of devices)
- Support ad hoc interactions (no Kerberos)
- Light on resources (possibly no asymmetric crypto)
- Interoperability - different vendors
- Renewability
- Multicast security
- Key Predistribution?

What is KPD?

- An (offline) TA and N nodes with unique IDs
- TA chooses P secrets $\mathbb{R}$
- Node A is pre-loaded with k secrets $\mathbb{S}_A = F(\mathbb{R}, ID_A)$
- Node B is pre-loaded with k secrets $\mathbb{S}_B = F(\mathbb{R}, ID_B)$
- Nodes A and B can discover shared secret $K_{AB} = G(ID_B, \mathbb{S}_A) = G(ID_A, \mathbb{S}_B)$
- Only nodes A and B can discover K_{AB}

n-Secure KPD

- Pre-loaded keys in different nodes are *not* independent
- A finite number of *other* nodes can be compromised to reveal K_{AB}
- n-secure KPD resists compromises of up to n nodes
- KPDs are tradeoffs between security and complexity
    - Large $n \rightarrow$ large k
    - Different mechanisms of trade-off
    - Efficient KPD schemes $k = O(n)$

Extents of Compromise

- Attacker pools keys from many node with the purpose of determining shares secret between
  - Two nodes *i* and *j* (Attack 1)
  - Node *i* and TA (Attack 2)
- All *P* secrets (Attack 3)

Classes of KPDs

- Deterministic KPDs based in finite field arithmetic (Blom, Matsuhito)
- Attacks 1,2,3 have the same complexity
- Subset intersection schemes (matrix, Mitch, Dyer, Erdos et al)
- Attacks 1 to 3 increasingly complex
- Random KPDs - provide only probabilistic guarantees
- For example, n-secure with probability of failure 10^{-20}
- Most random KPDs are based on subset intersection
- Exception - Leighton and Micali (Scheme II)
- Attacks 1 to 3 increasingly complex

Probababilistic Guarantees are Good Enough!

- Even for determinsitic schemes the final shared secret has a finite number of bits
- What is the probability that an attacker can "guess" a 64-bit key? - more than 10^{-20}.
- Probabilistic guarantees are not bad as long as the probability of failure is small

Random KPDs

- Two basic types
- Leighton and Micali (scheme III) - based on repeated hashing of preloaded keys
- Random preloaded subsets (RPS) - a slight modification of subset intersection schemes
- TA has P keys, each node is given a subset of k keys
- In SI schemes the allocation is done in a deterministic fashion
- In RPS it is done either randomly (Eschenauer-Gligor, Chan-Perrig-Song, Liu-Ning) or psuedo-randomly (Pietro-Mancini-Mei, Ramkumar-Memon)
- Former methods need bandwidth overhead to determine share keys - psuedo-random methods provide an elegant solution by using a one-way function of node ID

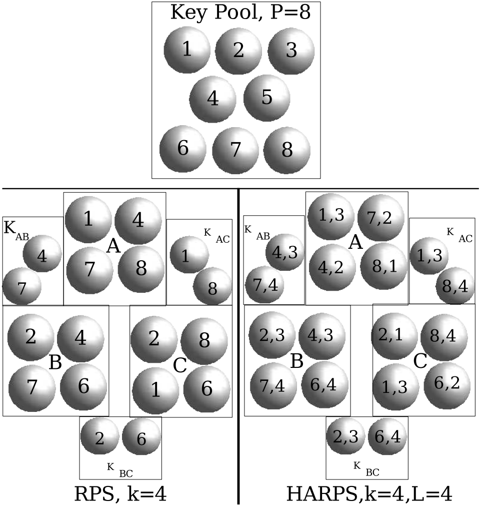
HAshed Random Preloaded Subsets

- Defined by three parameters, P, k, L
- TA chooses P secrets
- Each node gets a subset of the secrets (randomized by node ID)
- The preloaded keys are hashed repeatedly - a variable number of times
- Hash depths uniformly distributed between 1 and L (randomized by node ID)
- Shared secret based on maximum hash depths of the shared keys

HARPS, RPS and LM

- HARPS is a generalization of RPS and LM
- LM is HARPS with $P = k$
- RPS is HARPS with $L = 0$

Illustration of HARPS



Summary of Properties

- Efficient, $k = O(n)$
  - RPS, $k = O(n)$, LM, $k = O(n^3)$
  - RPS - $k = -\mathrm{e}\log(\mathrm{p})\mathrm{n}$, HARPS - $k = -\sqrt{\mathrm{e}}\log(p)n$
  - Theoretically, not possible to do better than $O(n)$
- Different threat models
  - How difficult is it to fool another node? (Attack 1)
  - To fool the TA? (Attack 2)
  - All random KPDs provide more resistance to Attack 2 (which is good)
  - HARPS does better than other random KPDs against Attack 1
  - And does very much better (by 2 orders of magnitude) against Attack 2.
  - Safe renewal with KPDs - need additional unique key or high resistance to attack 2

And More!

- Tree hierarchical extension (RPS - does not offer “seperation” of levels)
- Caters for seamless renewal
- The same preloaded secrets can also be used for
  - Broadcast authentication - equivalent to signature schemes
  - Targeted signatures / Designated verifiers...
  - Broadcast encryption - an efficient solution for node revocation
  - Discovery of group secrets
- Key Predistribution Infrastructure

KPI vs PKI

Feature

1. Deployment
2. Shared secret
3. Source Authentication
4. Non repudiation

PKI

1. tree hierarchical deployment of CAs
2. exchanging signed public keys
3. encrypting with private key
4. source authentication

KPI

1. tree hierarchical deployment of TAs
2. exchanging unique IDs
3. appending key based MACs
4. source authentication with trusted devices

KPI vs PKI

Feature	PKI	KPI
1. Revocation (1)	1. broadcasting revocation list	1. broadcasting revocation list
2. Revocation (2)	2. none	2. broadcasting revocation secret
3. Automatic revocation	3. expiry of certificate	3. periodic renewal
4. Seamless renewal	4. possible	4. possible with some loss of security
5. Broadcast Encryption	5. not possible	5. possible by TA and peers
6. Choosing Public keys	6. not possible	6. possible