

# Safe Renewal of a Random KPS for Trusted Devices

Mahalingam Ramkumar  
Mississippi State University

# Introduction

- Application setting
  - Applications involving *mutual co-operation* of *resource constrained* devices – eg AHNs
- Devices need to be *trusted*
  - Primary requirement: *hardware protection* for
    - *read-proofing* of secrets
    - *tamper-resistance*
- For long-lived security
  - secrets have to be *renewed periodically*

# Introduction

- The KDS - HARPS
  - **H**Ashed **R**andom **P**reloaded **S**ubsets
  - a random key pre-distribution scheme (KPS)
- Assurances (from technology) and security policies for *safe* renewal of secrets
  - Assurances
    - Read-proofing / Tamper-resistance
  - Security policies
    - Use of a password, additional unique secret, rest encryption
    - DOWN (decrypt only when necessary policy)

# Key Distribution Schemes (KDS)

- A method of distributing secrets (to each node)
  - the KDS secrets are used for mutual authentication
  - Authentication
    - Discovery of shared secret
    - Verification of claimed identities
- Type of KDS
  - Symmetric NS protocol (Kerberos),
  - PKI,
  - Key pre-distribution (KPS)

# Key Pre-distribution Schemes (KPS)

- Trade-off between security and complexity
- A trusted authority (TA), and  $N$  nodes with unique IDs
  - TA chooses a set of  $P$  secrets  $R$
  - Distributes  $k$  secrets to each node
  - Secrets  $S_A$  assigned to a node  $A$  is a function of the  $R$  and the unique ID  $A$  of the node - or  $S_A = f(A, R)$
  - Nodes  $A$  and  $B$  discover a unique shared secret  $K_{AB} = g_A(S_A, B) = g_B(S_B, A)$
  - TA's involvement is not needed for mutual authentication

# KPS vs KDS

- Kerberos / PKI
  - secrets distributed to (or chosen by) each node are *independent*.
- KPS – secrets are *not* independent
  - All secrets derived from TA's secrets **R**
  - Compromise of secrets from a **finite** number of nodes may lead to compromise of **all** secrets
  - **n**-secure KPS can resist compromise of **n** nodes
  - Usually **n** is linear in **k**.
    - Security-complexity tradeoff.

# Random KPS

- $n$ -secureness is not an adequate description of *random* KPSs
- A probability associated with failure.
  - $n$ -secure with probability of failure  $p$
  - $(n,p)$ -secure
    - Compromise of  $n$  nodes results in compromise of **shared** secrets with a probability  $p$
  - Even  $(n,p)$ -secure is not an adequate description

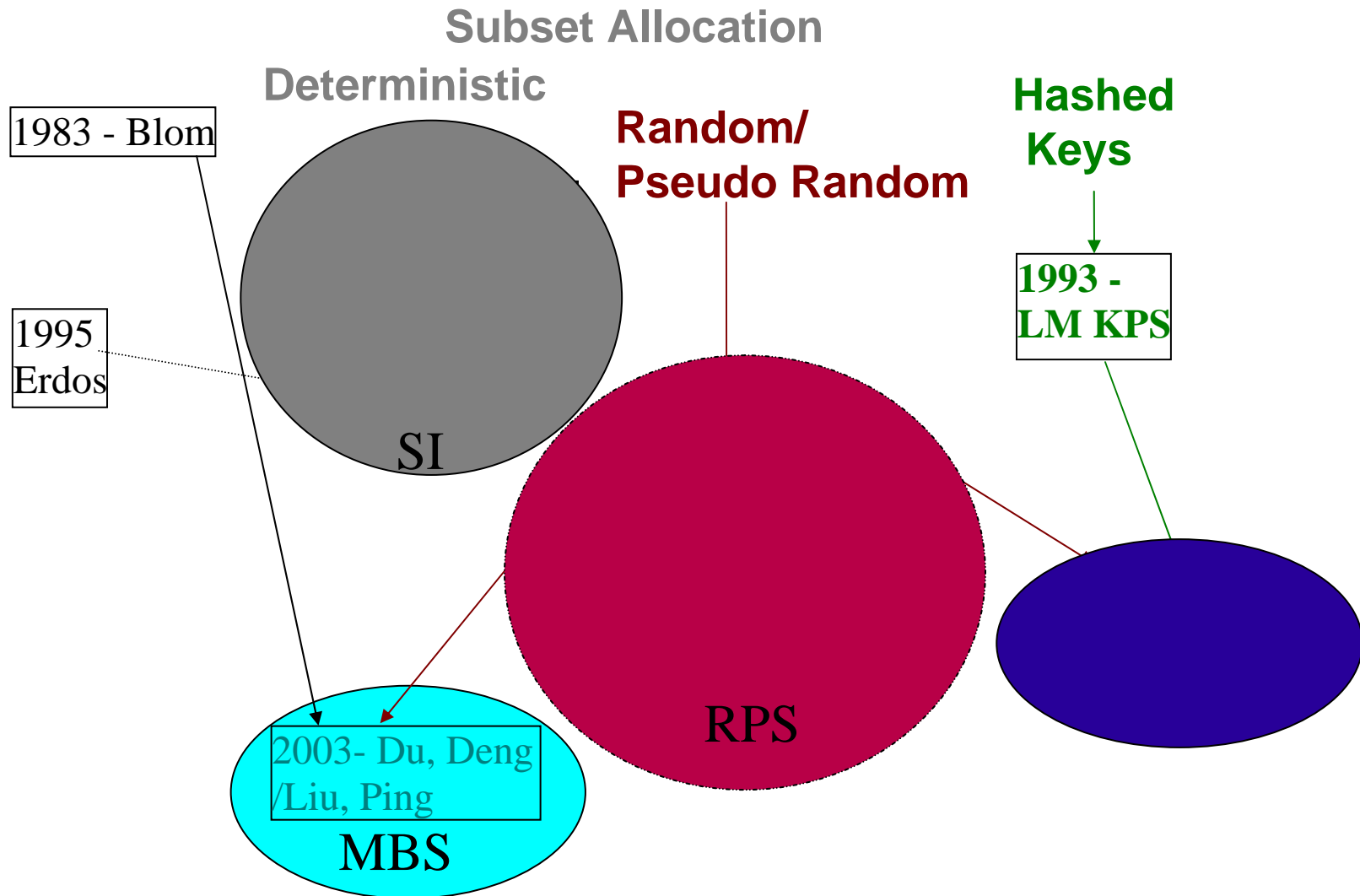
# Eavesdropping vs Synthesis

- $((n_e, p_e), (n_s, p_s))$  – secure
  - Secrets from  $n_e$  nodes need to be exposed for discovering **shared** secrets with probability  $p_e$ 
    - Enables attacker to impersonate nodes for purposes of fooling *other* nodes (but not the TA)
    - Eavesdropping attack
  - Secrets from  $n_s$  nodes need to be exposed for discovering **all** secrets of a node with probability  $p_e$ 
    - Enables attacker to fool even the TA
    - Synthesis attack

# Deterministic vs Random KPS

- For deterministic KPS eavesdropping attack and synthesis attack have the **same** complexity
- For random KPS **synthesis** attack is much more difficult
- This property is useful for **periodic renewal**
- *For renewal, attacker needs to fool the TA!*

# Brief History of Random KPS



# HARPS

- HARPS is a generalization of LM-KPS and RPS
- Hashed random preloaded subsets
- $(P,k,L)$ -HARPS
  - TA chooses  $P$  independent secrets
  - Each node receives a random subset of  $k$  secrets
  - Secrets provided to each node are hashed many times (a random number of times between 1 and  $L$ )

# HARPS vs RPS / LM

- All random KPS (except MBS) do **not** need multiplication
  - just a block cipher / hash function.
- For LM  $k \leq O(n^3)\log(1/p)$
- For RPS  $k = n e \log(1/p)$
- For HARPS  $k = n \sqrt{e} \log(1/p)$

# Node Synthesis

- For LM / RPS node synthesis is
  - an *order of magnitude* more difficult (number of nodes that need to be compromised) than eavesdropping attack
- For HARPS
  - it is **2-3 orders** of magnitude higher

# Assurances and Policies

- What are some “reasonable” assurances?
  - PUFs
  - Partial tamper resistance
- What are some security policies that can be useful?
  - Additional password for renewal
  - Additional stored secret used exclusively for renewal
  - DOWN policy

# PUFs

- Physical Unclonable Functions
  - Based on accidental uniqueness of delays in each fabricated chip
  - Provides a secret (or a one-way function) which cannot be exposed by tampering
  - However, the secret may be *weak* (low entropy)
- The low entropy secret could be used in addition to KPS secrets for renewal

# Forcing Synthesis With PUFs

- To brute force the weak secret the attackers need to
  - expose **all** KPS secrets in a node
  - without destroying the device
- Or a synthesis attack is necessary to expose all secrets by tampering with **other** devices
- Prohibitively expensive for HARPS.

# Forcing Synthesis With Assurance of Partial Tamper Resistance

- An assurance that only a fraction of the secrets can be exposed by tampering with a node
- A synthesis attack is necessary to expose **other** secrets.
- Again prohibitively expensive for HARPS.
- Either PUFs or partial TR is sufficient to force attacker to perform synthesis attack

# Other Security Policies

- Rest encryption using password / biometric signals
  - Eliminates stolen devices from being used for exposing secrets
- Password / Unique additional secret for renewal
  - Attacker has to tamper with the device to expose secret
  - After that, it is not possible to brute-force weak secret (in case PUFs are used)
  - Restricted to synthesizing nodes for which attacker knows the unique additional secret
    - Have to synthesize nodes with a much higher probability

# DOWN

- Decrypt only when necessary policy
- Developments in technology for TR indicate that it is possible to protect secrets while the device is
  - **in use** – using active sensors
  - **at rest** – using PUFs
- The most vulnerable period is the **transition** period
- DOWN seeks to **eliminate** explicit transition

# DOWN

- All cryptographic operations have some inherent **atomicity**
- Example, RSA – only one bit of private key is needed at any point in time
- With DOWN
  - Secrets are always encrypted
  - Decrypted *as and when necessary*
  - At any point in time not more than one secret or one part of secret is exposed (in RAM)

# Implications of DOWN

- Implications
  - Non volatile memory(NVM) - where secrets are stored encrypted - does not need any protection
  - Only one secret can be exposed by tampering with a device
- Bodes very well for KPSs
  - $n$ -secure KPS is rendered  $nk$ -secure
  - $k$  can be increased arbitrarily as NVM complexity is not a crucial issue!

# DOWN Complexity

- Each elementary DOWN operation involves
  - Fetching of secret from NVM
  - Switching to a secure kernel mode
  - Decrypting the secret
  - Using it in a cryptographic calculation
  - Flushing RAM clear of “footprints”
- Limitation – number of DOWN operations
- For RPS / HARPS  $k$  can be increased without increasing DOWN complexity!

# HARPS with DOWN

- HARPS with  $(P=15000, k=1000)$  and  $(P=60000, k=2000)$  have the same DOWN complexity
  - 67 DOWN operations ( $k^2/P = 67$ ) – average number of shared secrets
- For compromise probability of *one in a trillion*
  - 21,000 / 84,000 nodes need to be compromised for **eavesdropping** attack
  - 5 million / 20 million for **synthesis** attack!

# Conclusions

- HARPS can be rendered arbitrarily secure even for reasonable levels of complexity
- In evolving application scenarios TR is a necessary evil.
- When used in conjunction with reasonable assurances of TR and sensible security policies, the security of HARPS can be increased dramatically.