Architecture for Protecting Critical Secrets in Microprocessors

Lee, R.B.; Kwan, P.C.S.; McGregor, J.P.; Dwoskin, J.; Zhenghong Wang;

ISCA '05. Proceedings. 32nd International Symposium on Computer Architecture, 2005 04-08 June 2005 Page(s):2 - 13

Presented by Ahmed Mujuthaba

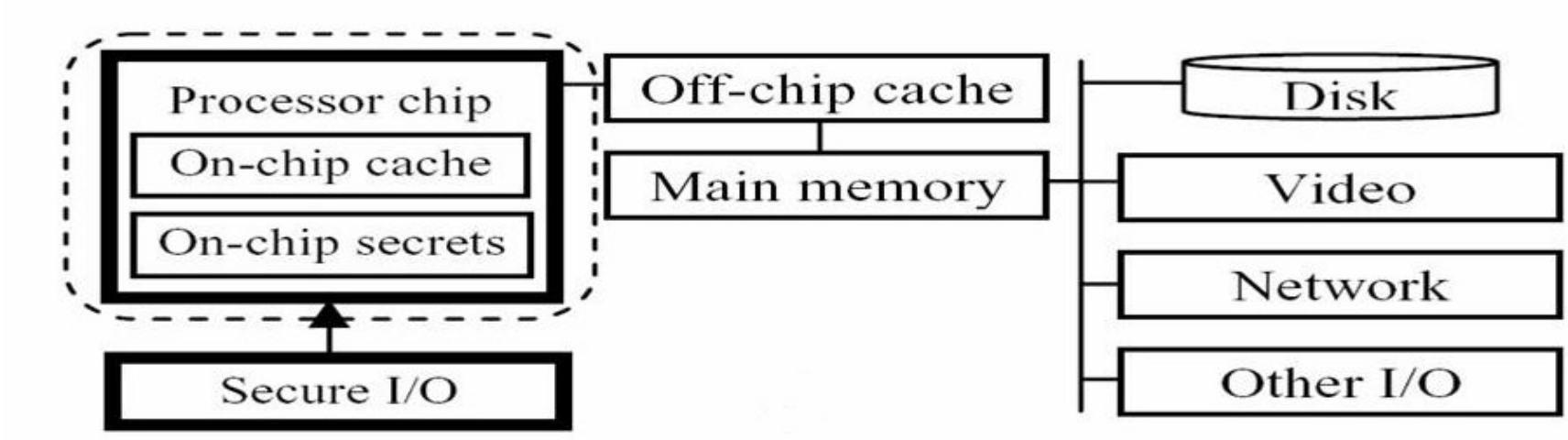
“Secret Protected” Architecture

- Critical secret to be protected:
  Cryptographic Keys.
- No factory installed device secrets.
- Decoupling of user secrets from device.
- Minimal addition to the general-purpose processor architecture.
- Only use symmetric master keys.

Keeping the secret safe in the face of:

- Statically & dynamically injected hostile code.
- Modification of software binaries in memory or disk
- Physical attacks such as probing of the external buses.

Trust model



Physical trust boundary – outlined in bold.

Device Master Key

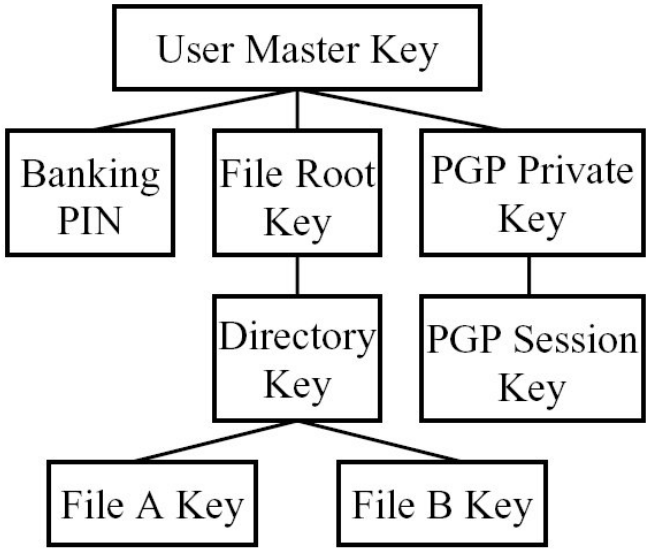
- Device Master Key installed when TSM is installed.
- Special secure (non-volatile) register in the processor to store Device Master Key.
- No processor instruction allows reading of the Device Master Key register.
- Device Master Key can be reset – but have to reinstall TSM as well.

User Master Key

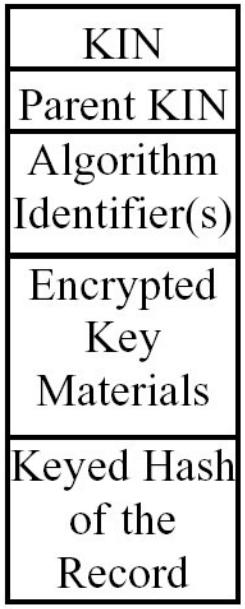
- User Master Key is associated with the user and not with the device.
- User Master Key generated from secret(s) possessed by user.
- Special secure register in the processor to store User Master Key for the duration of session.
- Only the TSM can access the User Master Key.

Key Chain

- Keys are hierarchically encrypted to form a key chain.
- Only a leaf key can be used to encrypt user data.
- All keys except the User Master Key can be in a publicly accessible (untrusted) repositories.



Key Chain



Key Record

What’s good about the paper

- Other existing architectures rely on a permanent factory installed secret.
- Trust is derived from uniqueness and provability of this secret.
- In SP-Architecture trust is derived from user secret.
- Hence trust is portable.

Possible performance issue?

- During installation of the TSM it is hashed using the Device Master Key installation. Hashes are generated per instruction cache line and stored inline with the code.

  When instruction is fetched in to the on-chip cache integrity of the code is verified by the processor using the hashes. After the integrity check the hashes are replaced by NO-OPs.

- The authors do not address the performance issues that may arise due to that many NO-OPs in the instruction stream.

Problem with Secure I/O method

- The authors propose the following as part of the secure I/O method:

  “After the user presses the Authenticate button, the platform switches the keyboard to a secure mode and begins diverting all keystrokes to the processor directly. Security against software attacks for this input path is provided by encryption from the keyboard to the processor. The processor’s Secure I/O Logic unit decrypts the keystrokes”

- The main problem with the above is the need for a key in the keyboard, since the keyboard cannot encrypt without one.
- Which key does the keyboard need?
- If the keyboard has the Device Master Key, the question of how it is installed and issue of it being compromised arise.
- If the keyboard and the Secure I/O unit share their own symmetric key, how is it installed and how secure is it?
- Solutions?

A Solution to the Secure I/O problem

- One method to solve this problem is to install a public key derived from the Device Master Key in the keyboard during installation.
- Device Master Key will be safe even if the keyboard is compromised.
- However this goes against one of the aims of the authors, namely not to use expensive public/private key pairs.

Question

- If such systems become cheaply available will you consider encrypting/signing your emails and files on a regular basis?
- If not, why not?