Protect Sensitive Data with Ada Keystore

Stéphane Carrez  FOSDEM 2020
Why Ada Keystore?

• Store sensitive parameters of an Ada server:
  – Database connection passwords
  – API secret keys

• Tool to store sensitive information:
  – Passwords and accounts
  – Encrypted documents without exposing their keys

https://github.com/stcarrez/ada-keystore
Getting a secret (1)

- Open keystore with simple password (simple but less secure : password in code!)

```ada
    with Keystore.Files;

    function Get_Secret return String is
        WS : Keystore.Files.Wallet_File;
        Pass : Keystore.Secret_Key := Keystore.Create ("...");
    begin
        WS.Open (Pass, "secure.akt");
        return WS.Get ("api-key");
    end Get_Secret;
```

https://github.com/stcarrez/ada-keystore
with Keystore.GPG;
with Keystore.Passwords.GPG;
with Keystore.Files;
function Get_Secret return String is
  WS   : Keystore.Files.Wallet_File;
  Ctx  : Keystore.Passwords.GPG.Context_Type;
begin
  Keystore.GPG.Open (WS, Ctx, "secure.akt");
  return WS.Get ("api-key");
end Get_Secret;

• Open keystore protected with GPG
Ada Keystore: a secure storage

- **AKT**
- **Ada Keystore**
- **GPG protected key**
- **User password**
- **Keystore**
- **Encrypted Data**

[https://github.com/stcarrez/ada-keystore](https://github.com/stcarrez/ada-keystore)
Ada Keystore: with multiple keys

User password

GPG protected key

Master key block

Directory key
Data info key
Data key key
Signature key

Directory keys

Data keys

Encrypted Data

Data fragment
Data fragment
Data fragment

Data fragment
Data fragment

https://github.com/stcarrez/ada-keystore
Ada Keystore: with blocks (unchained)

### Header block

<table>
<thead>
<tr>
<th>Ada (1815-12-10) (1852-11-27)</th>
<th>UUID</th>
<th>Storage ID</th>
<th>HDR Data</th>
<th>Storage Info + HMAC</th>
<th>HMAC-256</th>
</tr>
</thead>
</table>

### Master key block

<table>
<thead>
<tr>
<th>HDR</th>
<th>Wallet info</th>
<th>UUID</th>
<th>Key slot 1</th>
<th>...</th>
<th>Key slot 7</th>
<th>HMAC-256</th>
</tr>
</thead>
</table>

### Directory block

<table>
<thead>
<tr>
<th>HDR</th>
<th>Directory Name Entry</th>
<th>...</th>
<th>Data block indexes &amp; keys</th>
<th>HMAC-256</th>
</tr>
</thead>
</table>

### Data block

<table>
<thead>
<tr>
<th>HDR</th>
<th>Data fragment info</th>
<th>HMAC-256</th>
<th>Data Fragment</th>
<th>HMAC-256</th>
</tr>
</thead>
</table>

https://github.com/stcarrez/ada-keystore
Ada Keystore: key protection

Wallet master key block

User password

Wallet master key slot N (512 bytes)

Wallet Header Key, wallet header IV, Block ID

Wallet Sign

https://github.com/stcarrez/ada-keystore
Ada Keystore: directory encryption

Directory block

Directory Key, directory IV, Block ID

Directory Sign

Per data fragment encryption keys

Name (N bytes)  Entry ID (4 bytes)  Entry ID (4 bytes)  Store ID (4 bytes)  Block Num (4 bytes)  IV (128 bits)  key (256 bits)
Ada Keystore: data encryption

Data block

- Data fragment info
- HMAC-256

Data Fragment

- AES-256
- CBC

HMAC-256

Data Key, IV, Block ID

Data Sign

Data entry descriptor

- Entry ID (4 bytes)
- Slot Size (2 bytes)
- Data Offset (8 bytes)
- HMAC (256 bits)

Data fragment Key, IV, Block ID

https://github.com/stcarrez/ada-keystore
Ada Keystore: directory decryption

Directory block

Directory Key, directory IV, Block ID

Directory Sign

AES-256 CBC

HMAC-256

NO, invalid block

YES, valid block

Per data fragment encryption keys

Name (N bytes) Entry ID (4 bytes) Entry ID (4 bytes) Store ID (4 bytes) Block Num (4 bytes) IV (128 bits) key (256 bits)

HMAC-256

https://github.com/stcarrez/ada-keystore
Ada Keystore: data decryption

Data block

Data fragment info

HMAC-256

Data Fragment

HMAC-256

Data Sign

Data Key, IV, Block ID

AES-256

CBC

HMAC-256

Data entry descriptor

Entry ID (4 bytes)

Slot Size (2 bytes)

Data Offset (8 bytes)

HMAC (256 bits)

Data fragment Key, IV, Block ID

AES-256

CBC

HMAC-256

Data Fragment

https://github.com/stcarrez/ada-keystore
Ada Keystore: more complexity

Insert data
Update data
Prevent security holes

Multi-task encryption/decryption

Thread safe API
Secure data erase

Protect keys

https://github.com/stcarrez/ada-keystore
Ada benefit : limited types

- Use limited record :
  - Encryption keys cannot be copied
  - Secret key content visible only to AES operations
- Use Ada.Finalization.Limited_Controlled :
  - Erase encryption keys when object is released

```ada
type Secret_Key (Length : Key_Length) is limited private;
```

Ada benefit : subtypes

- Use subtype to constraint index computation:
  - Detect index errors when they are computed not when we access the data

```ada
subtype Buffer_Size is Stream_Element_Offset range 0 .. 4064;
subtype Block_Index is Stream_Element_Offset range 1 .. 4064;
subtype Block_Type is Stream_Element_Array (Block_Index);
```

```ada
Key_Pos := Key_Header_Pos - Key_Slot_Size (Iterator.Key_Count);
--- better to raise Constraint_Error here

Buf.Data (Key_Pos .. Key_Pos + Key_Size - 1) := ...;
--- bounds still verified
```

https://github.com/stcarrez/ada-keystore
Ada benefit: private child package

- Use private child package:
  - Restrict scope of operations
  - Helps in refactoring decisions
Ada benefit: precondition

- Use Preconditions:
  - Nice for API constraints
  - Helpful to detect internal errors earlier

```ada
function Contains
(Container : in Wallet;
Name      : in String) return Boolean is abstract with
Pre'Class => Container.Is_Open;

procedure PutUnsigned_32
(Into   : in out Marshaller;
Value : in Interfaces.Unsigned_32) with
Pre => Into.Pos <= Block_Type'Last - 4;
```

https://github.com/stcarrez/ada-keystore
Ada benefit: postcondition

- Use Postconditions:
  - Nice for API clarification
  - More complex to write

```ada
function Get_Header
(From : in out Marshaller) return Unsigned_32 with
  Post => From.Pos = Block_Type'First + 3;

procedure Get_Keys
(From : in Key_Provider;
 Key : out Secret_Key;
 IV : out Secret_Key;
 Sign : out Secret_Key) is abstract with
 Post'Class => Key.Length = 32 and IV.Length = 16
  and Sign.Length = 32;
```

https://github.com/stcarrez/ada-keystore
Ada benefit: protected types

- Use protected types:
  - Concurrent access to keystore objects
  - Holds keystore internal data structures

```ada
protected type Wallet_Container is
  procedure Open (...);
  procedure Create (...);
  procedure Add (...);
  ...
end Wallet_Container;
```
Ada benefit: tasks

- Use tasks for encryption/decryption process:
  - parallelize encryption and decryption
  - allow to configure the number of tasks
  - pre-allocation of tasks
Ada Keystore Work Queues

Data Fragment #1  Data Fragment #2  Data Fragment #N

Executor Task #1

Prepare Work  Executor Queue

READ BLOCK  AES-256 CBC  HMAC 256

Sequence Queue

Collect Work

Executor Task #N

READ BLOCK  AES-256 CBC  HMAC 256

Work Pool

https://github.com/stcarrez/ada-keystore
Conclusion

- Writing a secure keystore is hard:
  - insert, update, remove data entries that look random
  - don’t leak secret keys
- Project succeeded thanks to Ada:
  - many errors prevented by the language
  - no buffer overflow, runtime error detection
- Secure storage made simple for Ada:
  - simple Ada API (Open, Create, Add, Get, Remove)

https://github.com/stcarrez/ada-keystore
Questions

https://github.com/stcarrez/ada-keystore