Effective Cryptography

What's Wrong With All These Crypto APIs?

Thorsten Groetker, CTO
Utimaco, Inc.

utimaco®

Outline

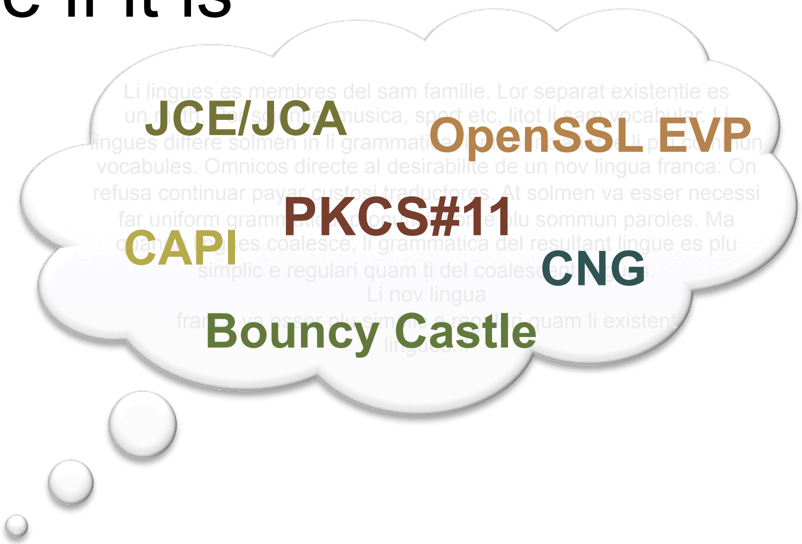
- What I mean by *Effective Cryptography*
- Crypto APIs
  - Security
  - Ease of Use
  - Runtime Performance
- Predictions
- CryptoScript in a Nutshell
- Outlook

Effective Cryptography

Definition in a Nutshell

Cryptography is effective if it is

1. Secure
2. Efficient
   a. Time to Result
   b. Performance



What's wrong with all these crypto APIs?
(Focused on Hardware Security Modules)

Problem #1: Security

PKCS#11

- Numerous key extraction attacks known
  - Jolyon Clulow "*On the Security of PKCS#11*"
  - Tookan project (e.g., "Attacking and Fixing PKCS#11 Security Tokens")
  - CVE entries (not necessarily sporting "*PKCS#11*" in the text)
  - … and so on
- Main culprits
  - Confusing set of mechanisms and attributes
    (it takes automated model checkers to determine secure configurations)
  - Functions broken into fine-grain operations
  - OS security, shared libraries, host debug hooks

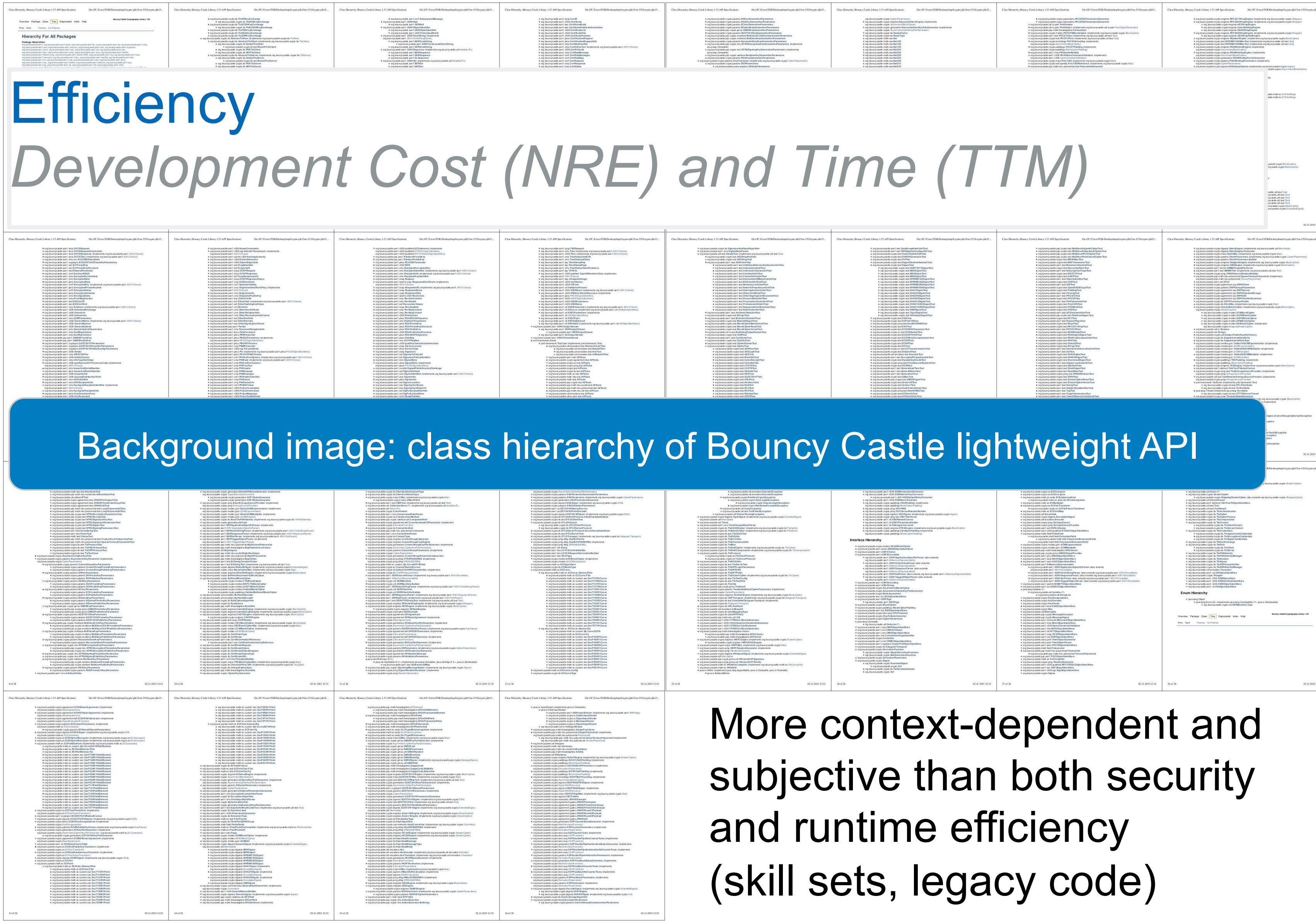
Problem #1: Security

Other host APIs

- Microsoft CryptoAPI (CAPI)
  - Exchange key pairs: encrypt and export session keys
  - Signature key pairs: sign messages
  - Exchange keys can be also used to encrypt/decrypt data ⇒ opens door to wrap-decrypt attacks
- JCE/JCA
  - Wrap-decrypt attacks possible unless prevented by underlying device
- Mixed APIs
  - Being able to access overlapping sets of keys from different APIs increases the attack surface and the likelihood for fixes to be bypassed

Efficiency

Development Cost (NRE) and Time (TTM)



Background image: class hierarchy of Bouncy Castle lightweight API

More context-dependent and subjective than both security and runtime efficiency (skill sets, legacy code) utimaco®

First Principles

- “*Simplicity is a prerequisite for reliability*.”
  And, hence, for security.

- Authentication should not be an afterthought.
  - Multi-factor
  - Multi-person (M-out-of-N) authentication

- Don’t forget about audit logging.

Performance Issues
Number Crunching vs Network

- Data transfers can easily become the dominating factor

  Server ↔ Cryptographic Service Provider ↔ Middleware/Network ↔ Network Appliance ↔ Driver ↔ HSM

- Your mileage may vary
  - Number of round-trip data transfers per function
  - Latency vs throughput
  - HSM load balancing

- Implement cryptographic functions as atomic HSM commands
  - It's faster
  - It's more secure

KMIP to the Rescue?

Batched Requests and Responses

The protocol contains a mechanism for sending batched requests and receiving the corresponding batched responses, to allow for higher throughput on operations that deal with a large number of entities, e. g., requesting dozens or hundreds of keys from a server at one time, and performing operations in a group. … A special ID Placeholder … is provided in KMIP to allow related requests in a batch to be pipelined.

[KMIP Protocol Use Guide]

☺ Addresses some performance issues

☹ Not suited as general crypto programming paradigm

Personal Prediction

- **Crypto Apps running within the secure perimeter of an HSM will become the norm.**
- Drivers include security, ease of use, performance, multi-tenancy, custom logging, portability, and cost.
- Firewalling, key binding (to app), app binding (to device), and strong authentication will become hard requirements.
- In a couple of years, users will start asking for standards.

Quick check: Attack surface comparison

- Crypto app running inside HSM w/ ± 5 ext. commands
- PKCS#11 host program w/ access to 50+ functions, 200+ mechanisms, and 50+ attributes.

From Embedded Software to Apps

Game Changer

Don't forget how dramatically

- an easy-to-use API
- combined with firewalling
- enabling 3rd party apps

can change an established market.

NOS

Symbian

Android

From Embedded Software to Apps

Game Changer

Don't forget how dramatically

- an easy-to-use API
- combined with firewalling
- enabling 3rd party apps

can change an established market.

Managed Language

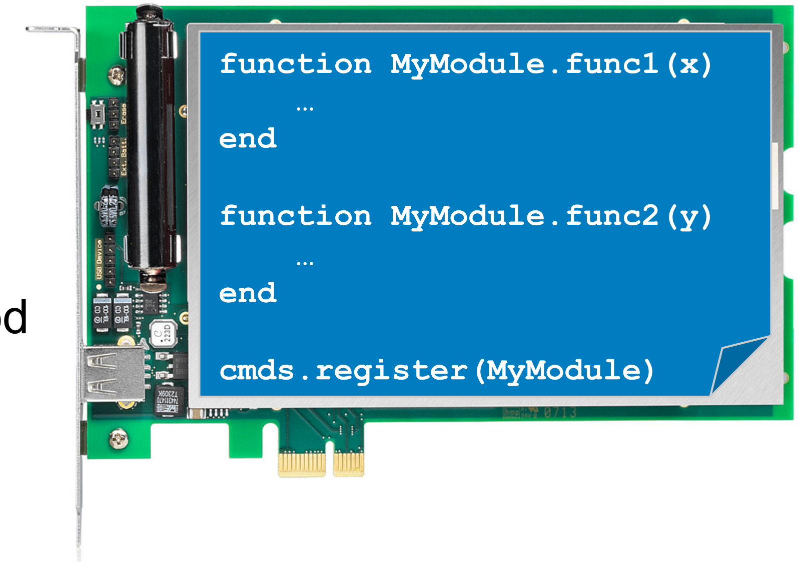
- Automatic garbage collection
- Firewalling, ease of use
- Device independent, portable

Introducing CryptoScript

Flow: easy as 1-2-3

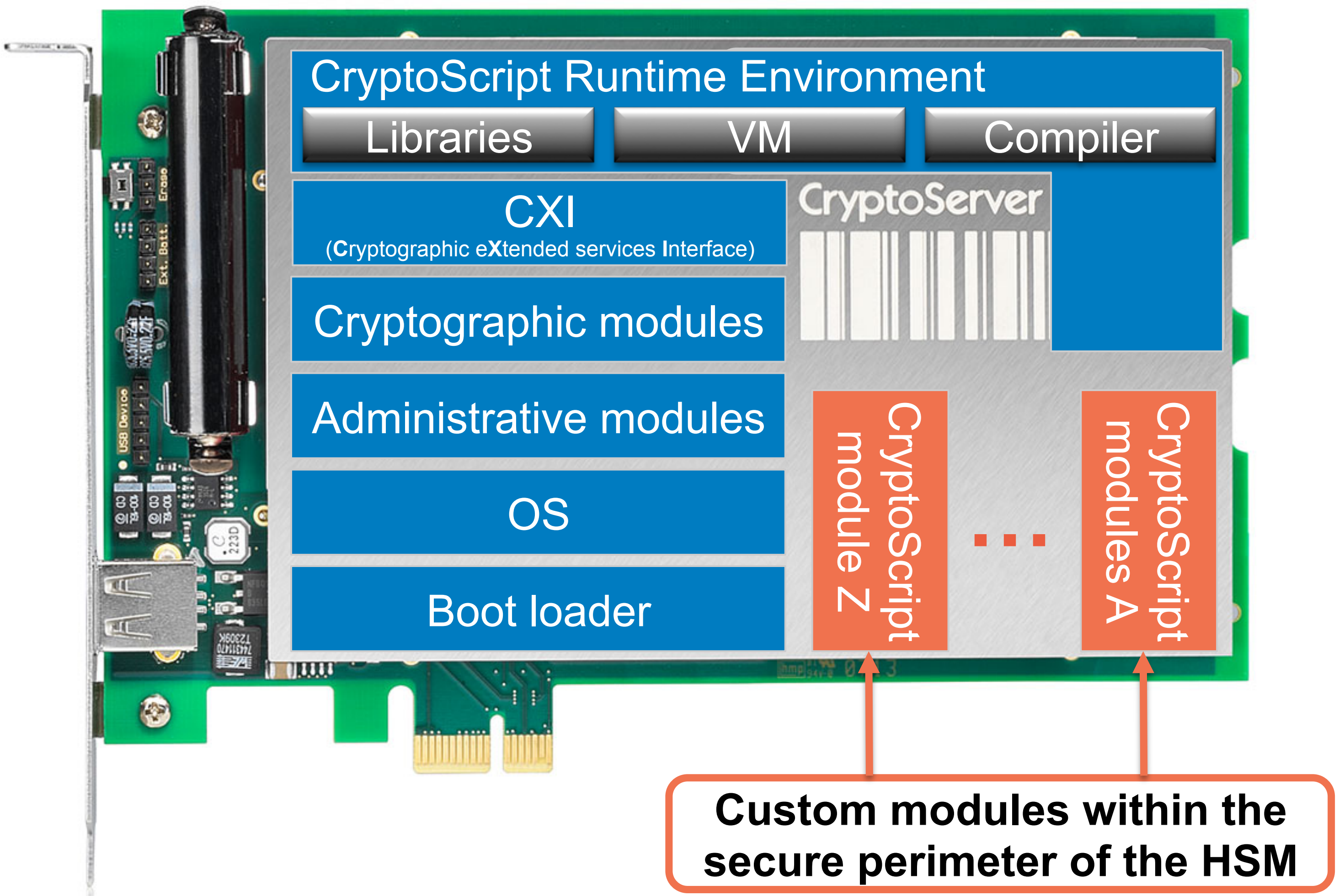
1. Write script

2. Load (signed) script
   - Automatically compiled under the hood and executed once, where it …
   - spawns threads and/or …
   - registers functions as commands



3. Invoke newly registered CryptoScript commands
   - From host application (C, C++, Java, C#)
   - From command line (host)
   - Cannot tell the difference to commands implemented in firmware

Introducing CryptoScript



CryptoScript Concept
Core Language

- Derived from Embedded Lua
  - Small, efficient, portable, MIT license
  - First class functions, support for OO design, automatic garbage collection
- Pared down by removing …
  - Application program interface, native debug I/F, aux lib, OS facilities, …
- Enhanced by adding …
  - Secure managed memory
  - Command handling, authentication, and secure messaging
  - Lua interface to CXI class hierarchy
    - Cryptography, arbitrary precision (modular) integer arithmetic
    - DB, pin-pad and smartcard access
  - Cryptographically secured debug interface

CryptoScript Concept

Secure Managed Memory

Managed Memory

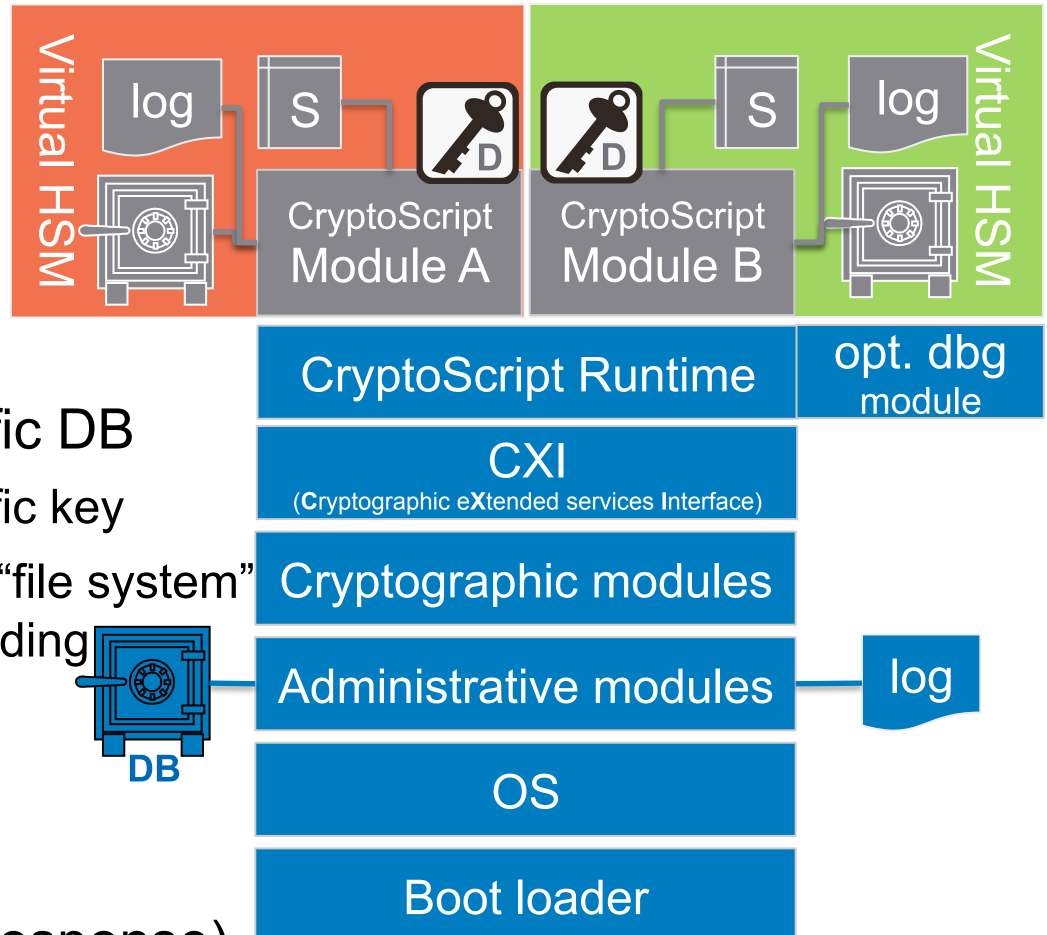
- No direct memory addressing
- No buffer/stack overflows

Optimized for HSM usage

- Low memory overhead and fragmentation
- Secure memory attribute
  - Objects stored in secure memory area (erased on alarm)
  - Attribute is inherited/propagated so that derived data is also located in secure memory

CryptoScript Concept
Virtual HSM

- Separate state/SMM (S)
- Separate audit logs
  - Contains FW and script info
  - Per-module log access key
- Optionally: module-specific DB
  - Encrypted w/ module-specific key
  - Keys, byte code, “registry”, “file system”
    ⇒ Strong key- and data-binding
  - Backup/restore supported
- No direct access to HSM file system and memory
- Opt. dbg key (challenge/response)



CryptoScript Concept

Main CryptoScript Classes

CXI	listKEYS(), generateKEY(), openKEY(), deleteKEY(), …
	hash(), encrypt(), decrypt(), sign(), verify(), …
KEY	access to key attributes (via associative array)
	derive(), copy(), wrap(), unwrap(), backup(), restore(), …
ATTR	collection of attributes (associative array), ± key template
	e.g., KEY_NAME, KEY_GROUP, …
MECH	mechanisms and parameters
	e.g., IV, CHAIN, …
BN	arbitrary precision integer, slices & concatenation, logic,
	(modular) arithmetic, random/primes, comparison, …

Symmetric encryption example

Pared-down example from R&D test suite

```
…
attr = ATTR.new();
attr.KEY_ALGO = "KEY_ALGO_AES";
attr.KEY_GROUP = "test";
list_of_keys = cxi:listKEYS( attr ); -- AES keys in group "test"

for _key_attr, key_attr in ipairs( list_of_keys ) do

    key = cxi:openKEY( key_attr, CXI.FLAG_KEY_VOLATILE );
    plain = BN.new("0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF");

    mech = MECH.new();
    mech.CHAIN = "CHAIN_CBC";
    mech.IV = "0123456789ABCDEF";

    cipher = cxi:encrypt( key, mech, plain );
    …
```

CryptoScript

Unique Combination of Benefits

- Secure
  - Compiled & executed within secure perimeter of HSM
  - Attack surface substantially reduced compared to host APIs
- Easy to use
  - No embedded SW skills/tools required
  - Development possible on simulator or HSM
- Fast
  - Single call to compiled CryptoScript function from server application
  - Cryptography based on highly optimized firmware / HW acceleration

CryptoScript

Outlook

- Email me for (draft) version of CryptoScript Reference Manual
- Concept → Early Access Program → General Availability
- Secure E2E communication: proprietary solution → SCP03?
- Open CryptoScript Initiative?

Thank You

Thorsten Groetker

thorsten.groetker@utimaco.com

Utimaco IS GmbH

Germanusstr. 4
52080 Aachen
Germany
Tel +49 241 1696 200
Fax +49 241 1696 199

Utimaco Inc.

475 Alberto Way Ste 120
Los Gatos, CA 95032
United States of America
Tel +1 408 395 6400
Fax +1 408 402 3598