Advanced Software Protection: Integration, Research, Exploitation

Bjorn De Sutter
bjorn.desutter@ugent.be

18 Oct 2016

Belgian OWASP Chapter Meeting

Aspire in a nutshell

NAGRA SafeNet gemalto security to be free

SafeNet use case

Gemalto use case

Nagravision use case

Software Protection Tool Flow

Protected SafeNet use case

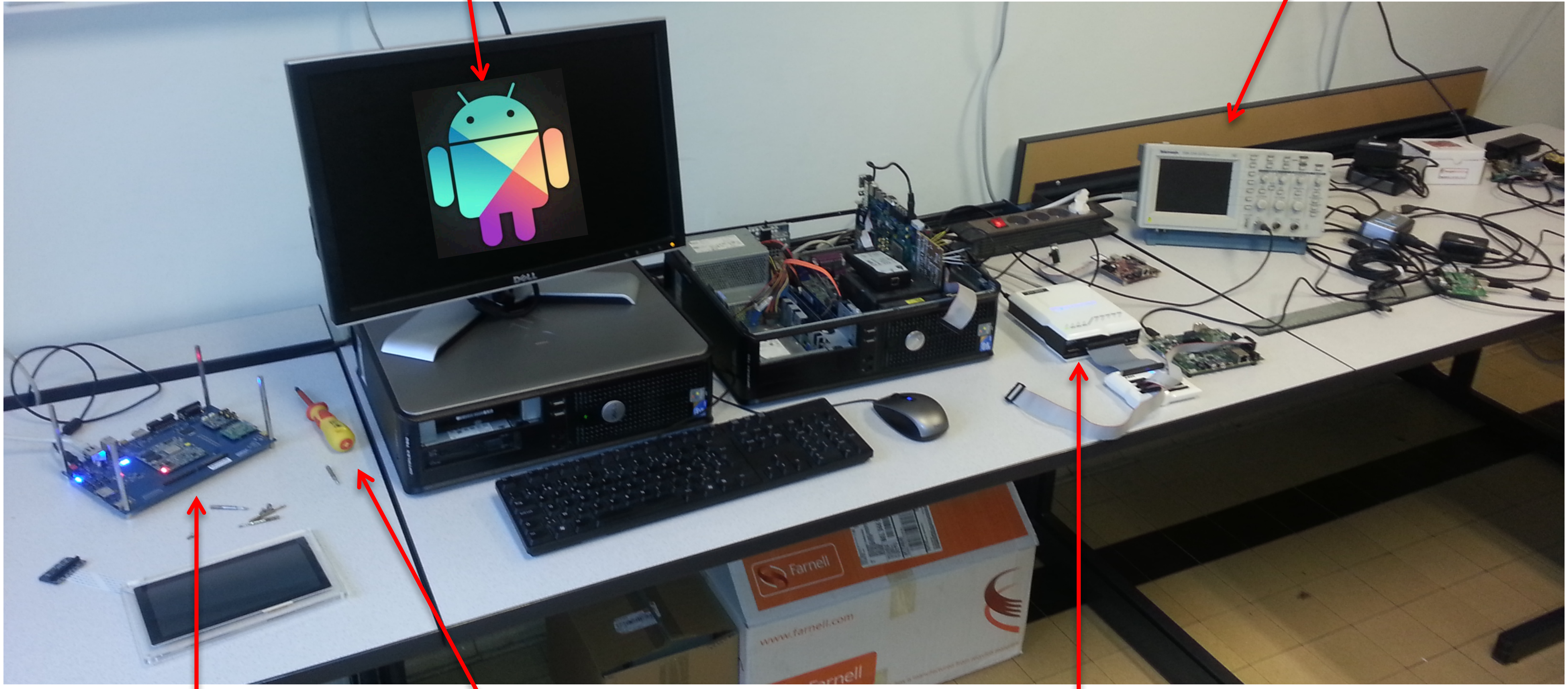
Protected Gemalto use case

Protected Nagravision use case

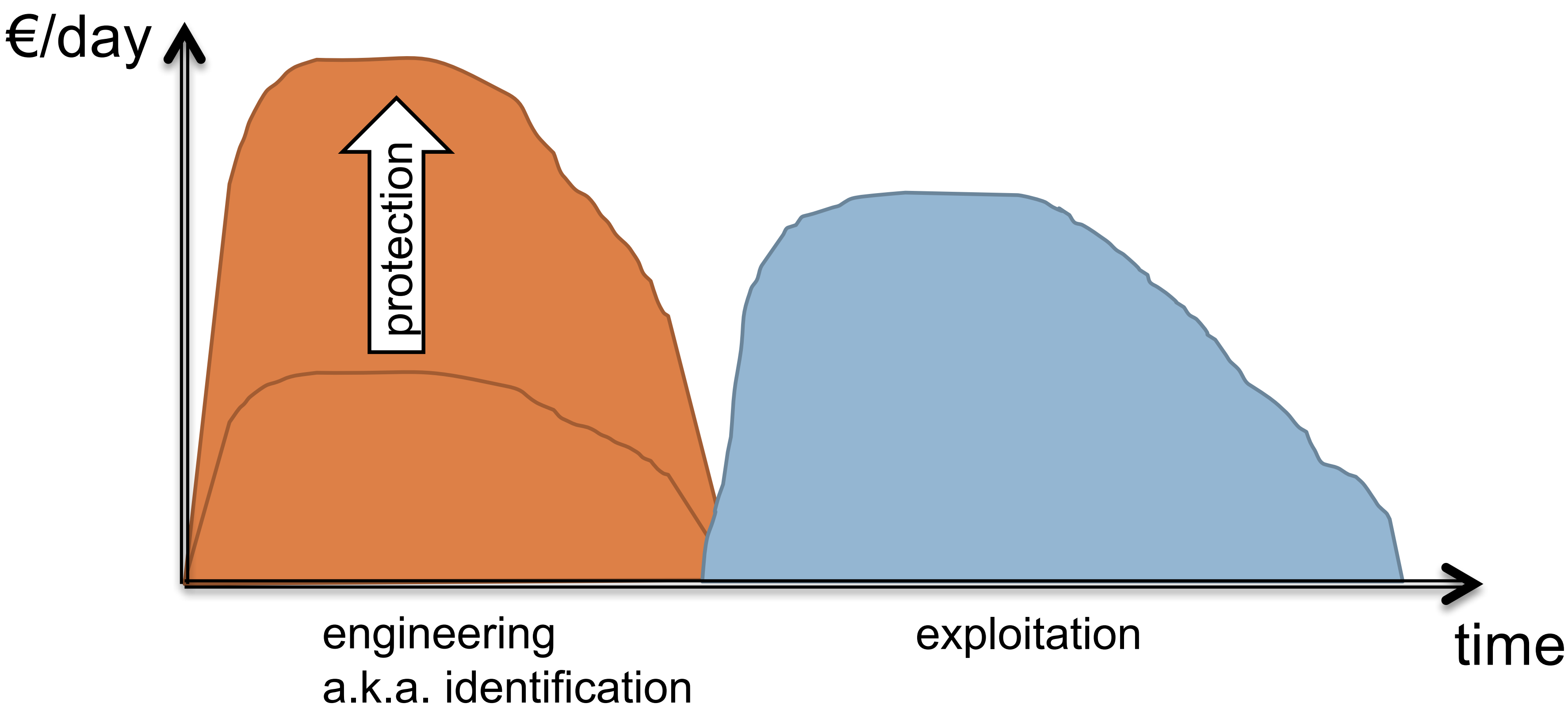
Data Hiding | Algorithm Hiding | Anti-Tampering | Remote Attestation | Renewability

Man-At-The-End (MATE) Attacks

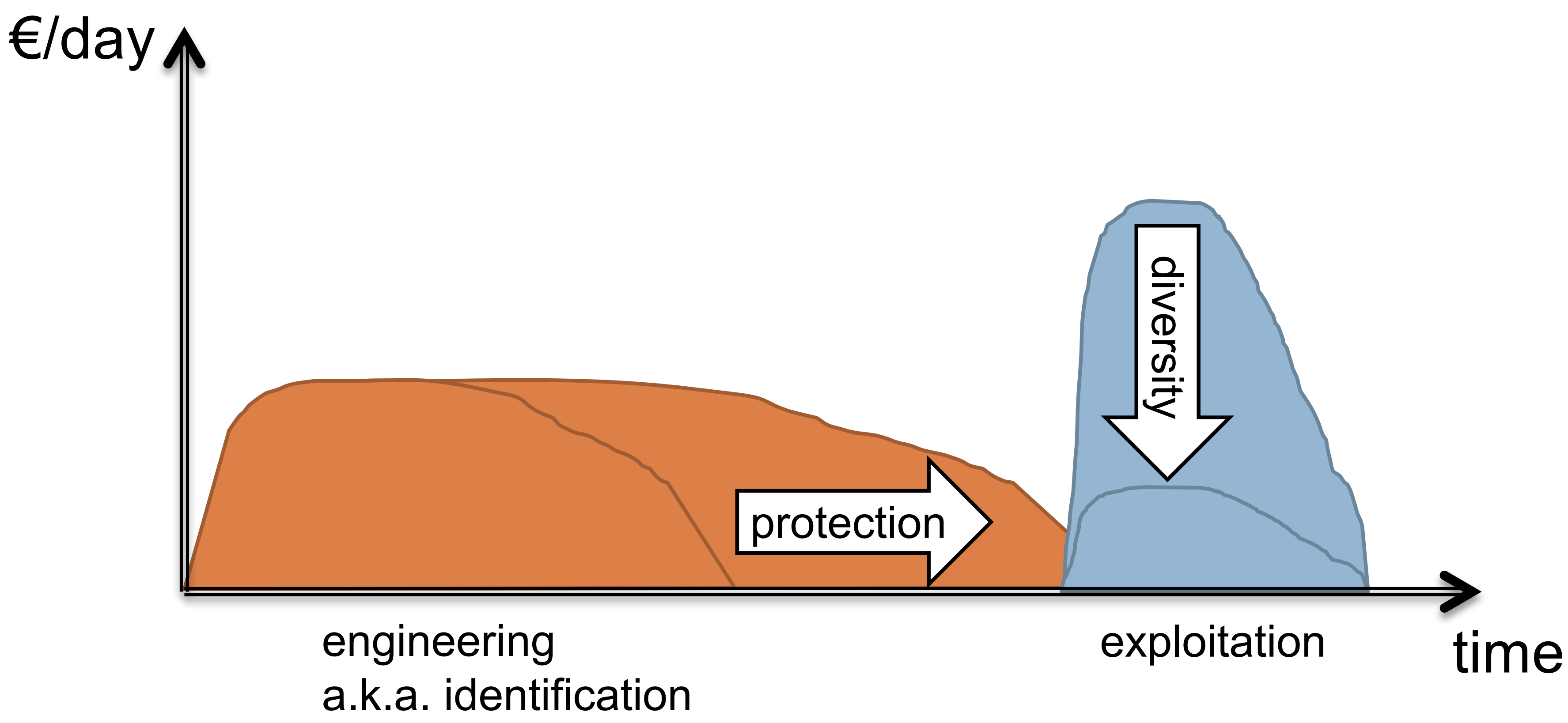
Man-At-The-End (MATE) Attacks



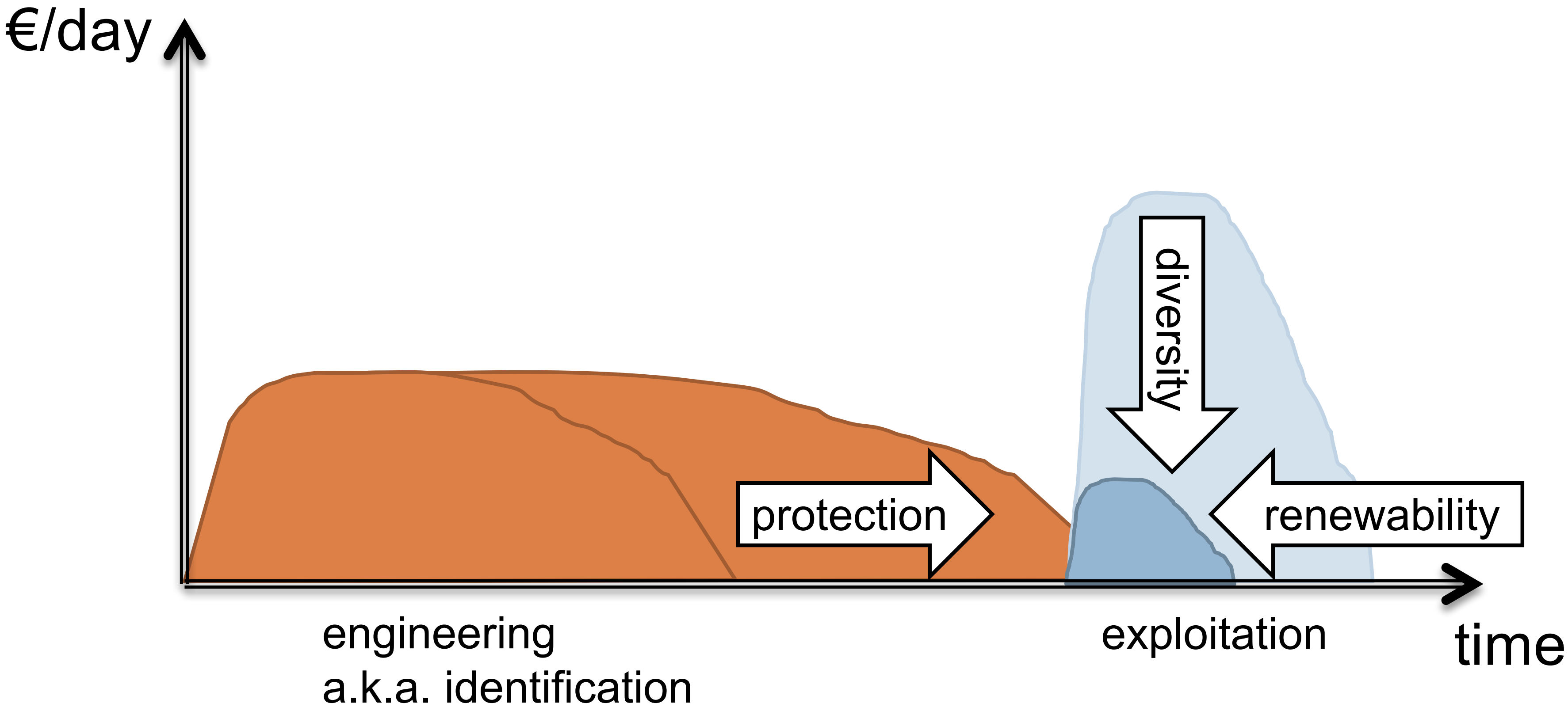
Economics of MATE attacks



Economics of MATE attacks



Economics of MATE attacks



Assets and security requirements

Asset category	Security Requirements	Examples of threats
Private data (keys, credentials, tokens, private info)	Confidentiality Privacy Integrity	Impersonation, illegitimate authorization Leaking sensitive data Forging licenses
Public data (keys, service info)	Integrity	Forging licenses
Unique data (tokens, keys, used IDs)	Confidentiality Integrity	Impersonation Service disruption, illegitimate access
Global data (crypto & app bootstrap keys)	Confidentiality Integrity	Build emulators Circumvent authentication verification
Traceable data/code (Watermarks, finger-prints, traceable keys)	Non-repudiation	Make identification impossible
Code (algorithms, protocols, security libs)	Confidentiality	Reverse engineering
Application execution (license checks & limitations, authentication & integrity verification, protocols)	Execution correctness Integrity	Circumvent security features (DRM) Out-of-context use, violating license terms

Aspire in a nutshell

NAGRA | SafeNet | gemalto security to be free

SafeNet use case → ASPIRE Framework → Protected SafeNet use case

Gemalto use case → ASPIRE Framework → Protected Gemalto use case

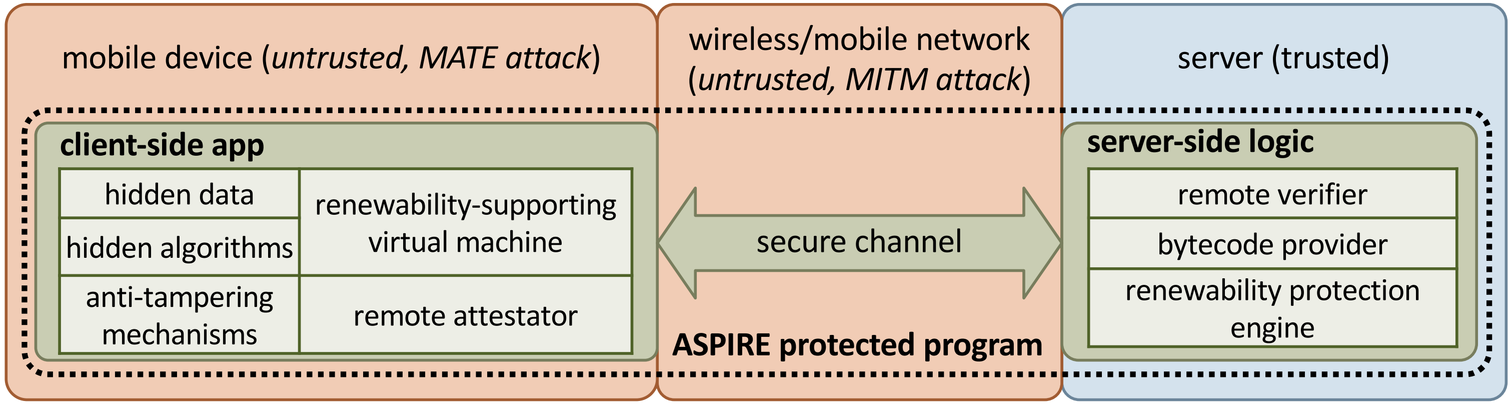
Nagravision use case → ASPIRE Framework → Protected Nagravision use case

ASPIRE Framework

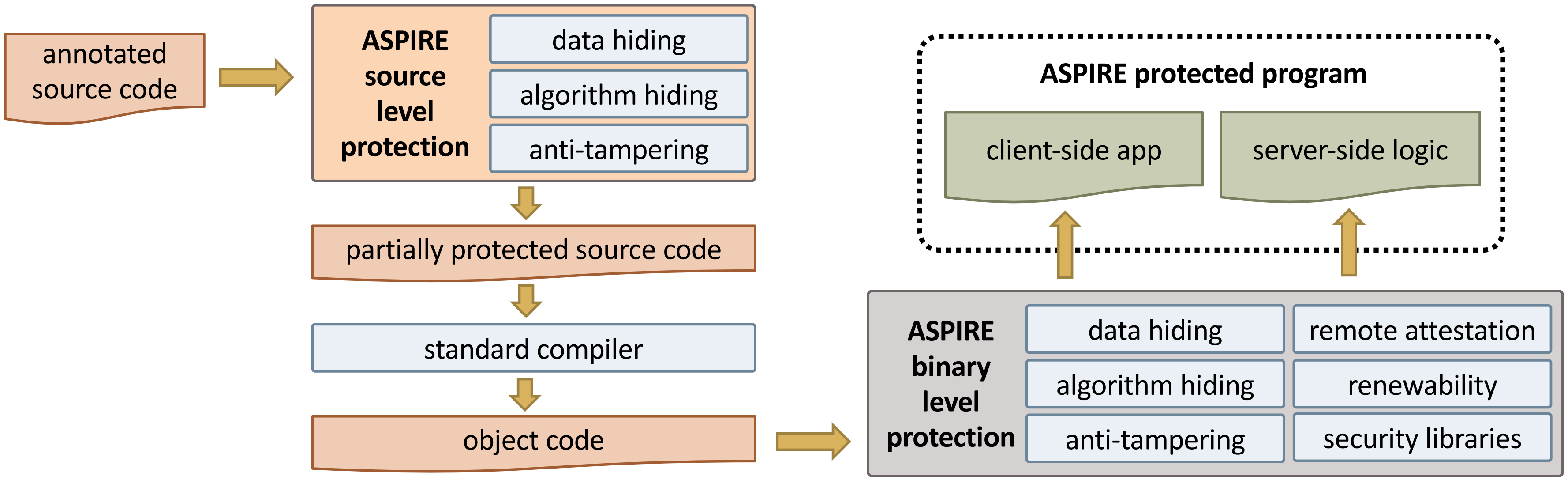
- **Decision Support System**
- **Software Protection Tool Chain**

Data Hiding | Algorithm Hiding | Anti-Tampering | Remote Attestation | Renewability

1. Reference architecture for protected mobile services

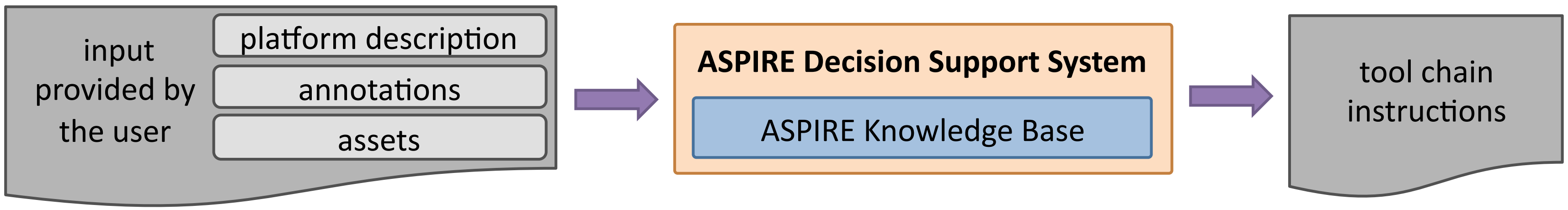


2. Software protection techniques and integrated plugin-based tool flow



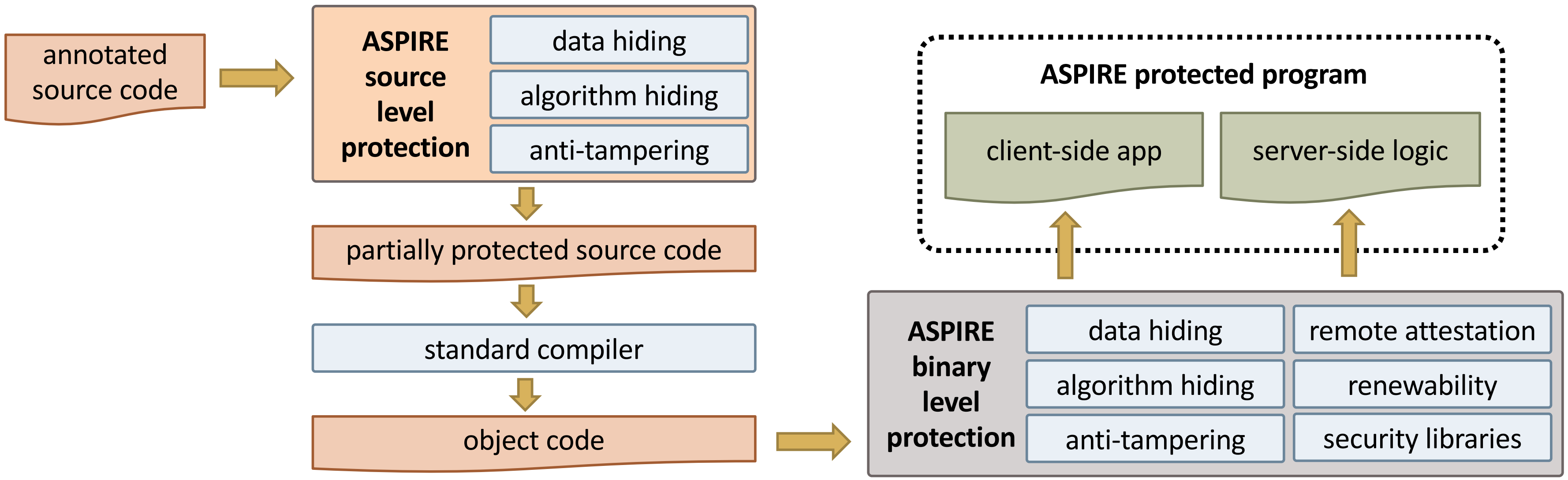
Aspire Goals

3. Decision Support System



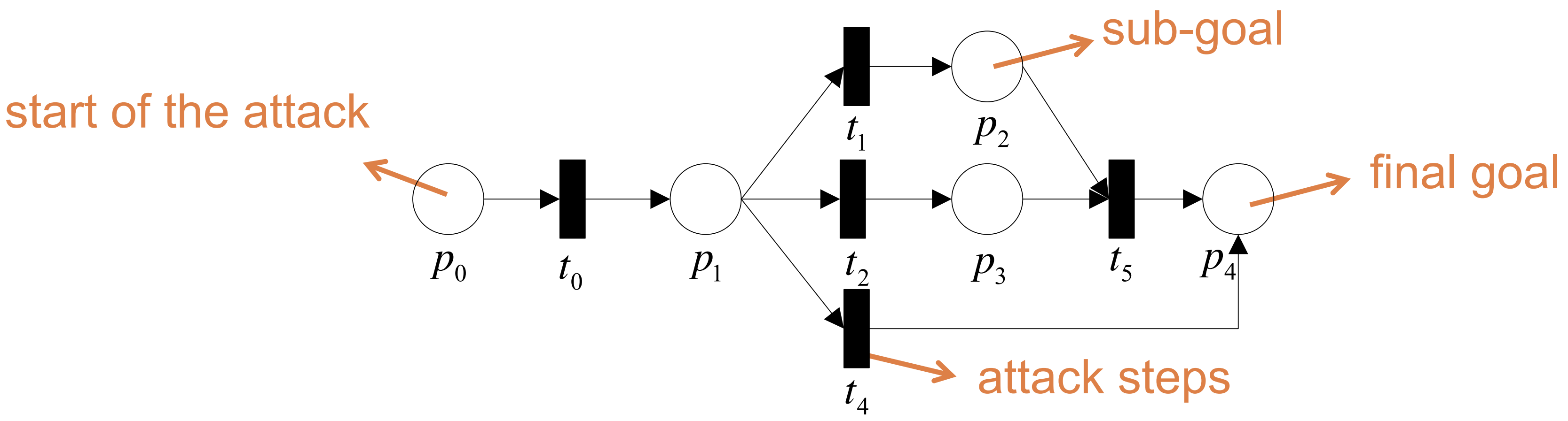
- attack models & evaluation methodology
- security metrics
- experiments with human subjects
- public challenge

2. Software protection techniques and integrated plugin-based tool flow



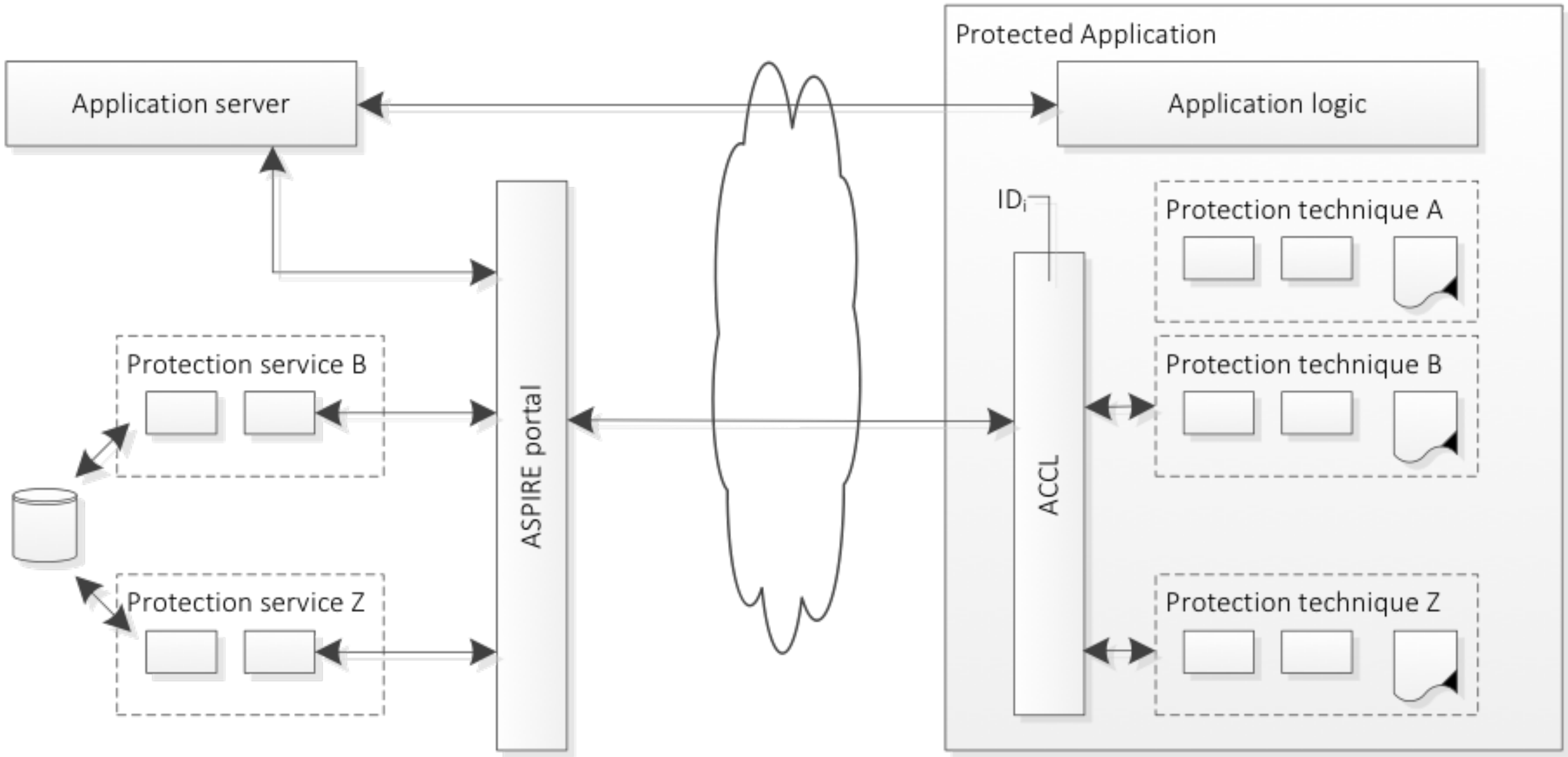
Part 1: Reference Architecture

- Cookbook for combining protections
- Why?



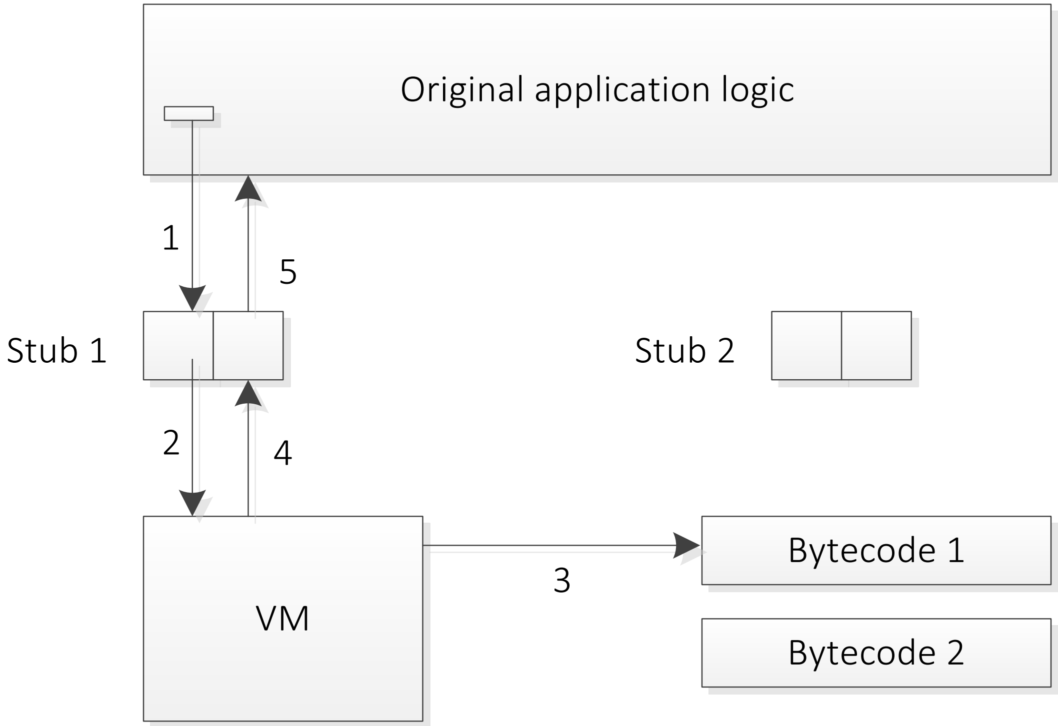
Part 1: Reference Architecture

- How to combine multiple protections?
  - How do the individual protections actually work?



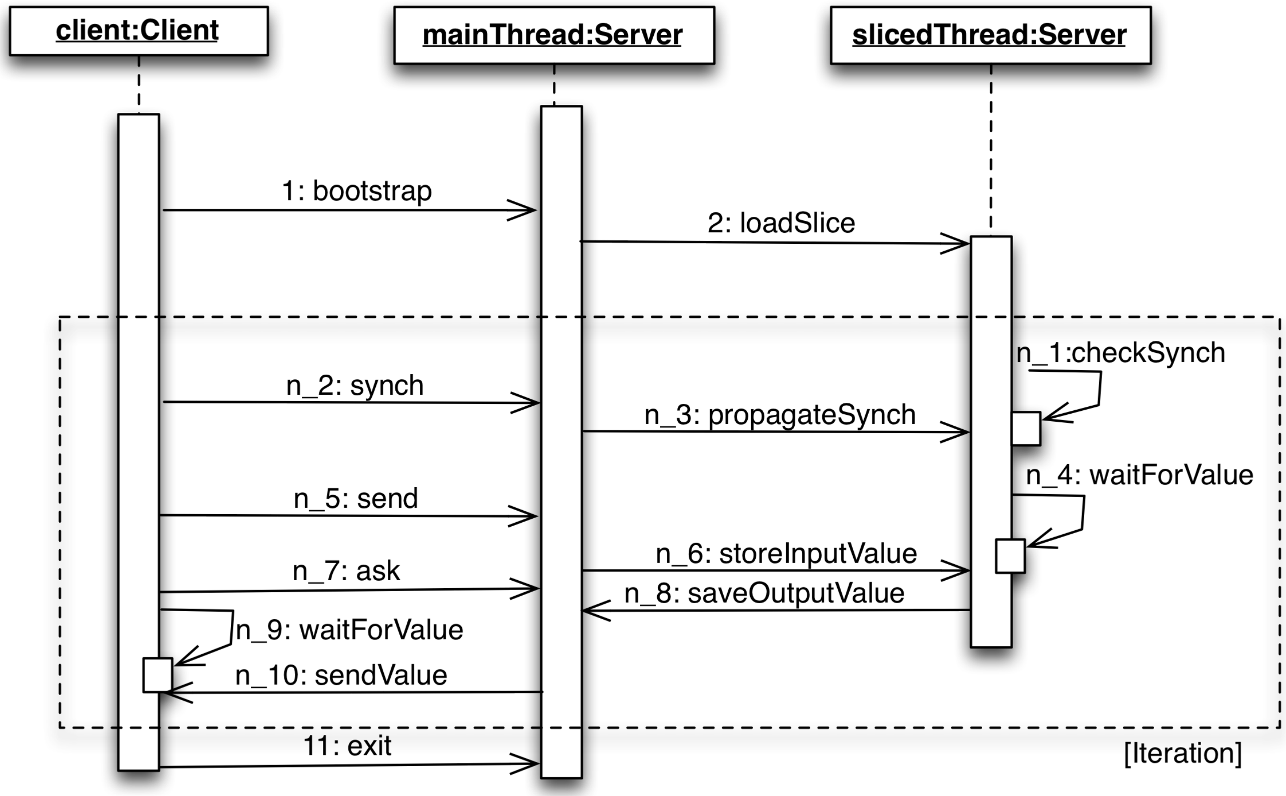
Part 1: Reference Architecture

- How to combine multiple protections?
  - How do the individual protections actually work?



Part 1: Reference Architecture

- How to combine multiple protections?
  - How do the individual protections actually work?



Part 1: Reference Architecture

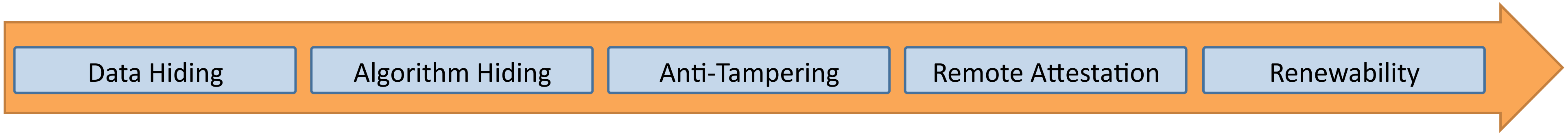
- How to combine multiple protections?
  - How do the individual protections actually work?

Data Hiding | Algorithm Hiding | Anti-Tampering | Remote Attestation | Renewability

- data obfuscations
- white box cryptography (static keys, dynamic keys, time-limited)
- diversified crypto libraries

Part 1: Reference Architecture

- How to combine multiple protections?
  - How do the individual protections actually work?



- control flow obfuscations
- multithreaded crypto
- instruction set virtualization
- code mobility
- self-debugging
- client-server code splitting

Part 1: Reference Architecture

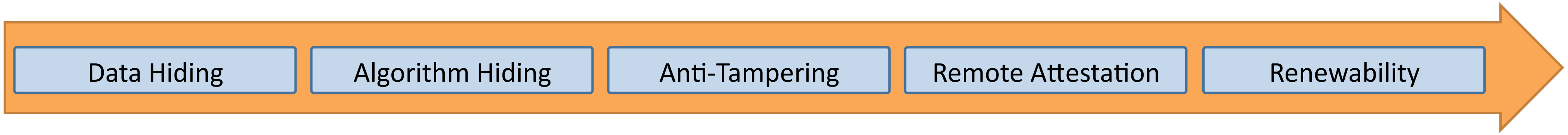
- How to combine multiple protections?
  - How do the individual protections actually work?

Data Hiding | Algorithm Hiding | Anti-Tampering | Remote Attestation | Renewability

- code guards
- static and dynamic remote attestation
- reaction mechanisms
- client-server code splitting

Part 1: Reference Architecture

- How to combine multiple protections?
  - How do the individual protections actually work?



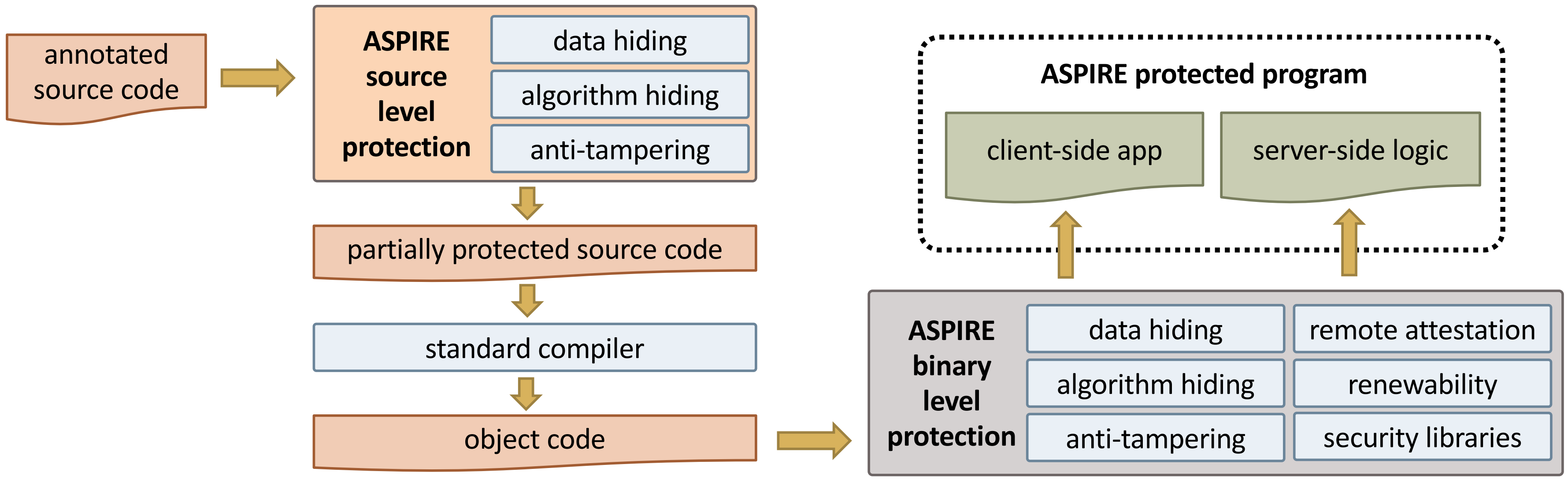
- native code diversification
- bytecode diversification
- renewable white-box crypto
- mobile code diversification
- renewable remote attestation

Part 1: Reference Architecture

- How to combine multiple protections?
  - How do the individual protections actually work?
  - How do the protections compose?
  - Do the protections share components?
  - If protections compose, are there phase-ordering issues?
  - Which protections/components need to be combined and how?
  - Where is 1 + 1 > 2 in terms of protection strength?
  - What is the combined impact on software development life cycle?

Part 2: ASPIRE Compiler Tool Chain

2. Software protection techniques and integrated **plugin-based** tool flow

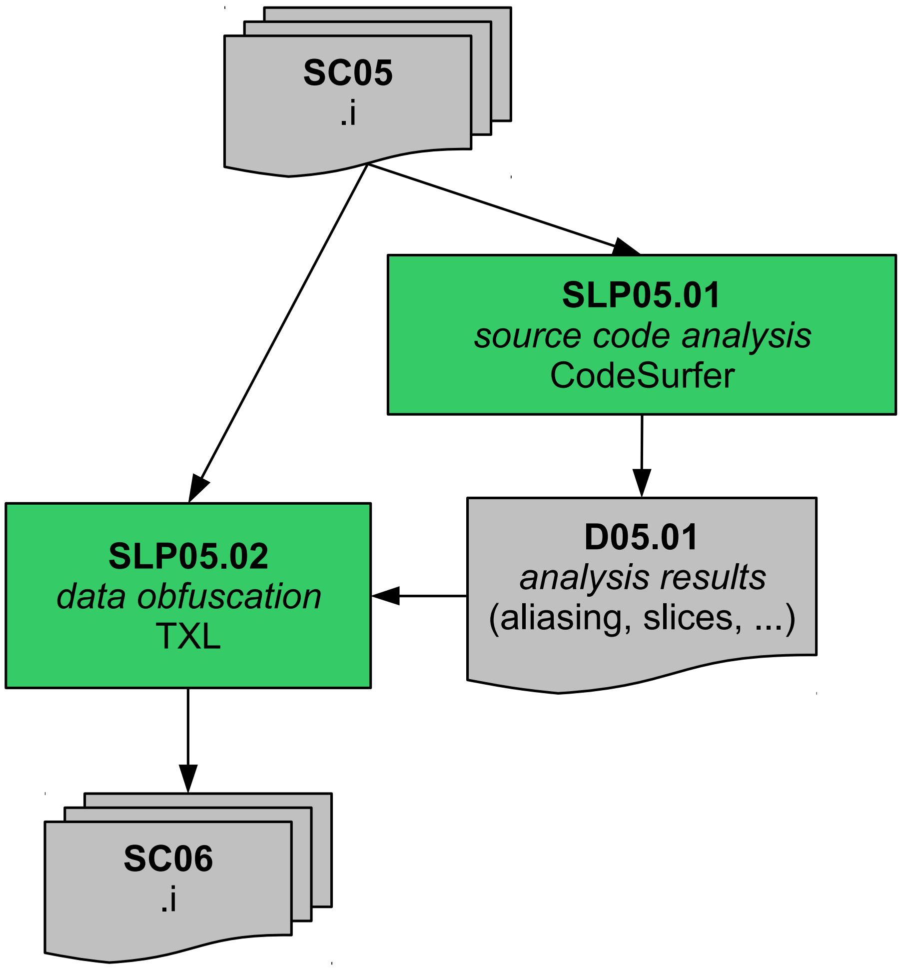


- Python – DoIt compiler flow
- JSON configuration scripts
- invokes chain of +/- independent tools
- TXL source code rewriting
- Diablo link-time binary rewriting

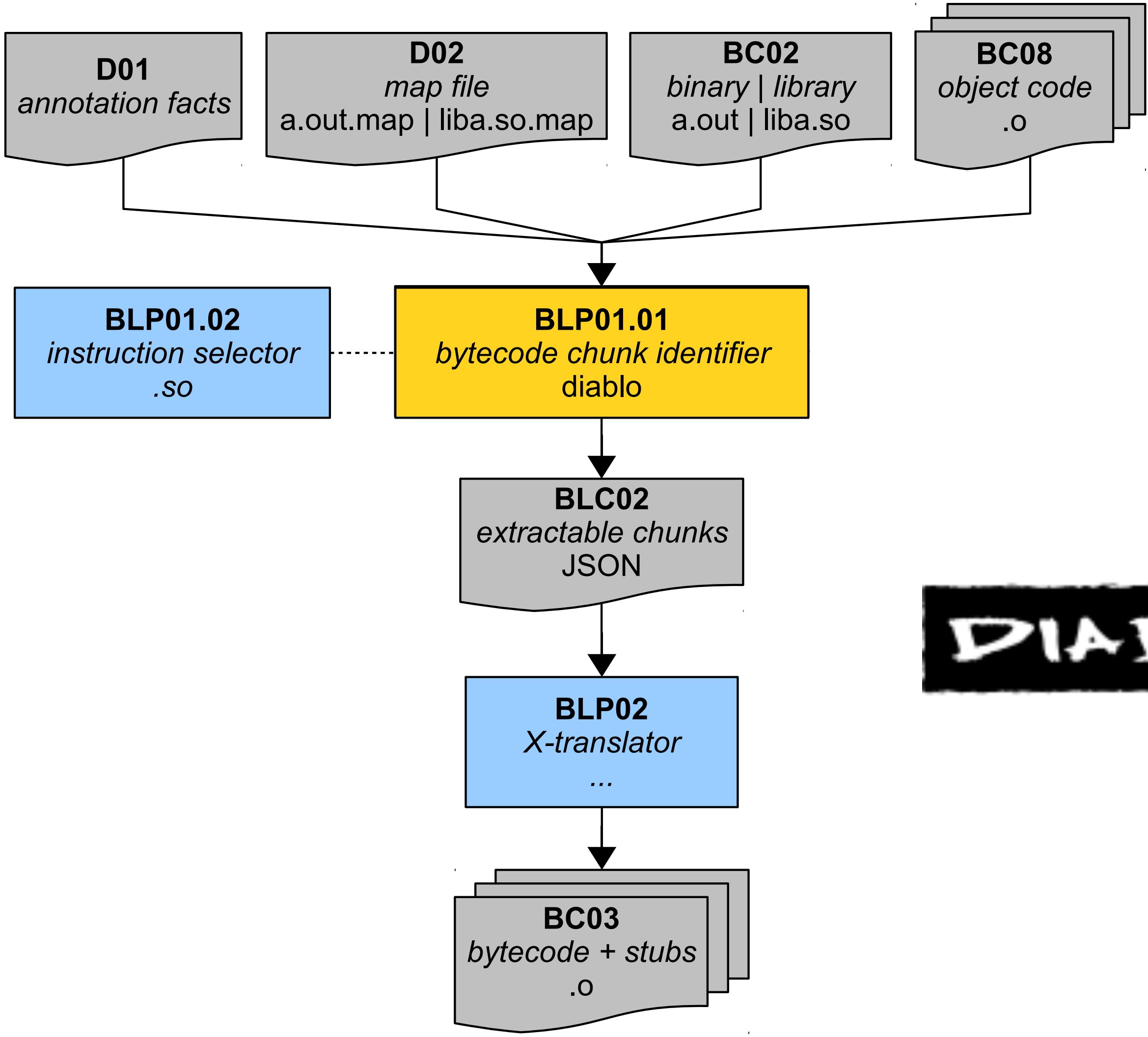
Source code annotations


```
static const char ciphertext[] __attribute__
  ((ASPIRE("protection(wbc,label(ExampleFixed),role(input),size(16))")))
   = { 0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07,
       0x08, 0x09, 0x0a, 0x0b, 0x0c, 0x0d, 0x0e, 0x0f };

static const char key[] __attribute__
  ((ASPIRE("protection(wbc,label(ExampleFixed),role(key),size(16))")))
   = { 0x00, 0x11, 0x22, 0x33, 0x44, 0x55, 0x66, 0x77,
       0x88, 0x99, 0xaa, 0xbb, 0xcc, 0xdd, 0xee, 0xff };

char plaintext[16] __attribute__
  ((ASPIRE("protection(wbc,label(ExampleFixed),role(output),size(16))")))
  ;

_Pragma ("ASPIRE begin protection(wbc,label(ExampleFixed),algorithm(aes),mode(ECB),operation(decrypt)")")
decrypt_aes_128(ciphertext, plaintext, key);
_Pragma("ASPIRE end");
```

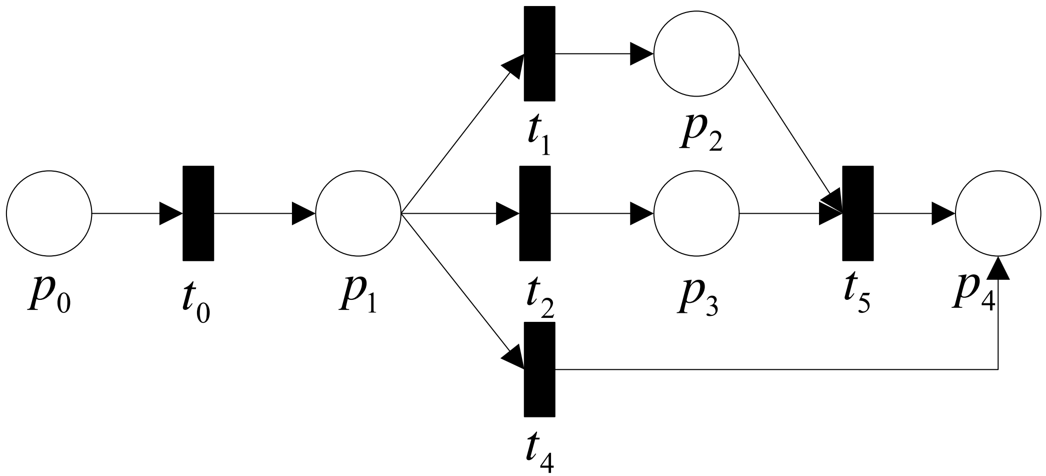
Source Code rewriting



Binary Code Rewriting



Part 3: Decision Support



- Knowledge Base
- Complexity & Resilience Metrics
- Protection Strength Evaluation Methodology
- Optimization strategies

Validation & Demonstration

- three real-world use cases
  - software license manager
  - one-time password generator
  - DRM protection

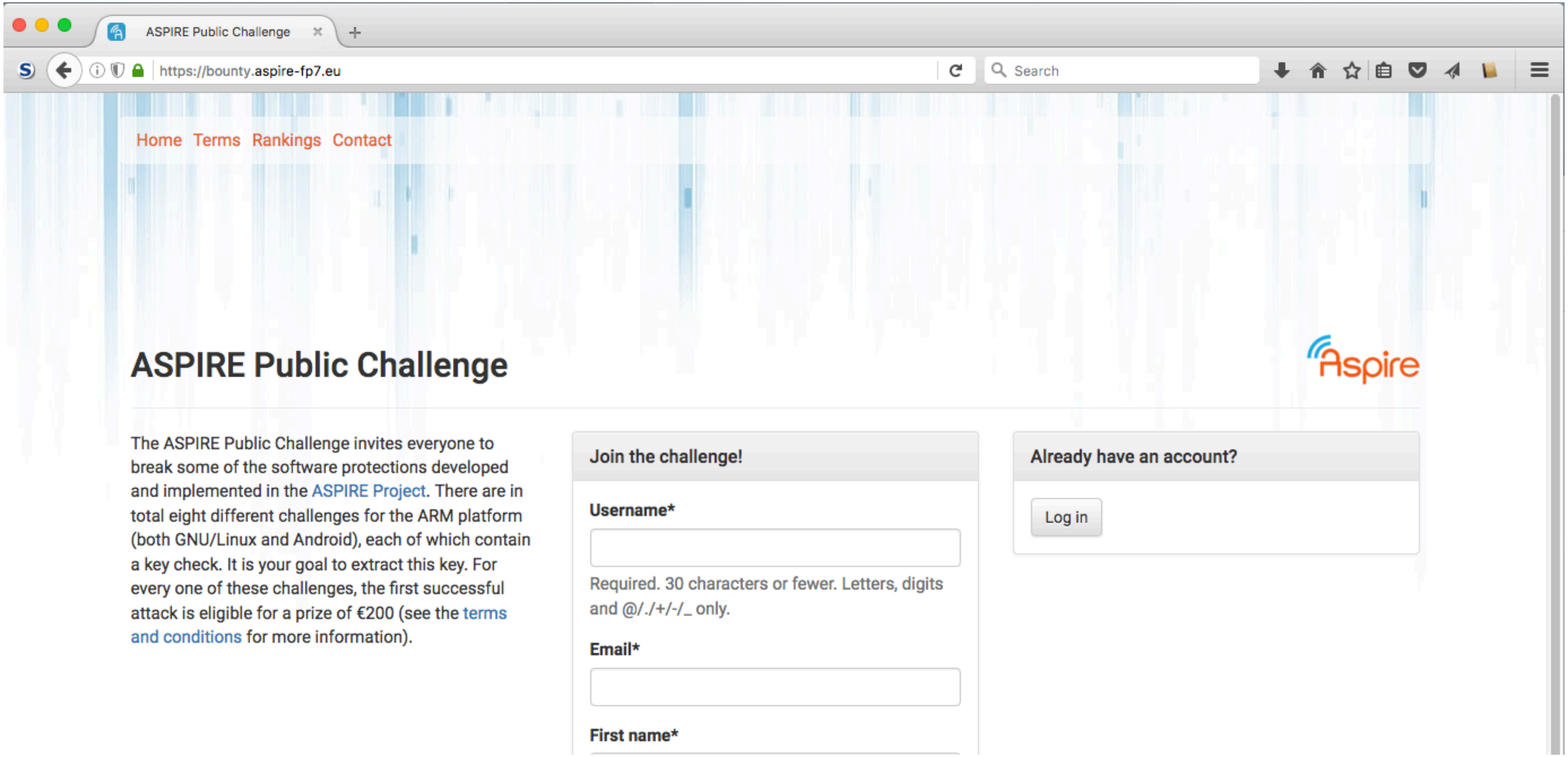
- security requirements from industry
  - functional requirements
  - non-functional requirements
  - assurance requirements

- dynamically linked Android 4.4 – ARMv7 libraries

- penetration tests by professional pen testers

Validation & Demonstration

- controlled experiments with academic hackers
- public challenge and bounties



More resources

- https://www.aspire-fp7.eu
  - papers
  - public reports
  - contact info

- https://github.com/aspire-fp7
- https://github.com/diablo-rewriter

- Youtube channel: ASPIRE-FP7 Software Protection Demonstration

Aspire Grant Agreement No 609734

The Aspire project has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement number 609734.

If you need further information, please contact the coordinator:
Bjorn De Sutter, Ghent University
Technologiepark-Zwijnaarde 15, B-9052 Gent, Belgium
Tel: +32 9 264 33 67 Fax: +32 9 264 35 94
Email: coordinator@aspire-fp7.eu
Website: https://www.aspire-fp7.eu