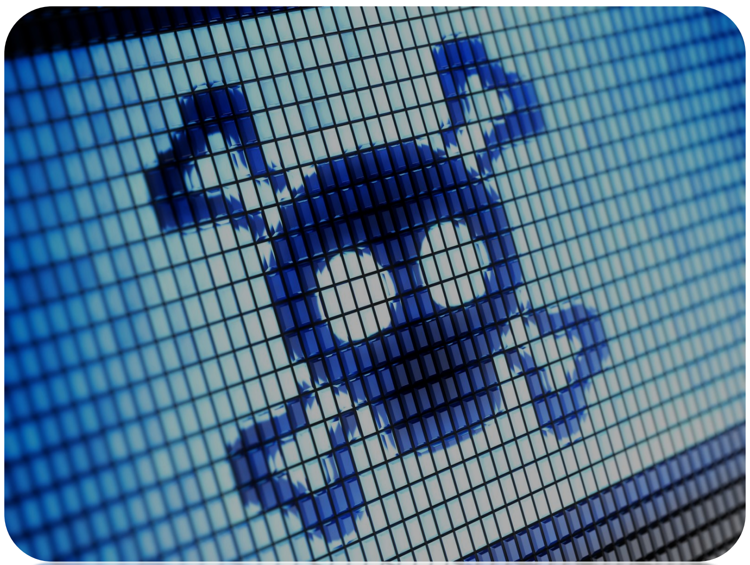
Soluções em Segurança da Informação

FORTUNA – Sorte e Azar

Dr. Roberto Gallo
SP, 2012-12-08
Diretor Executivo
Cientista Chefe
gallo@kryptus.com

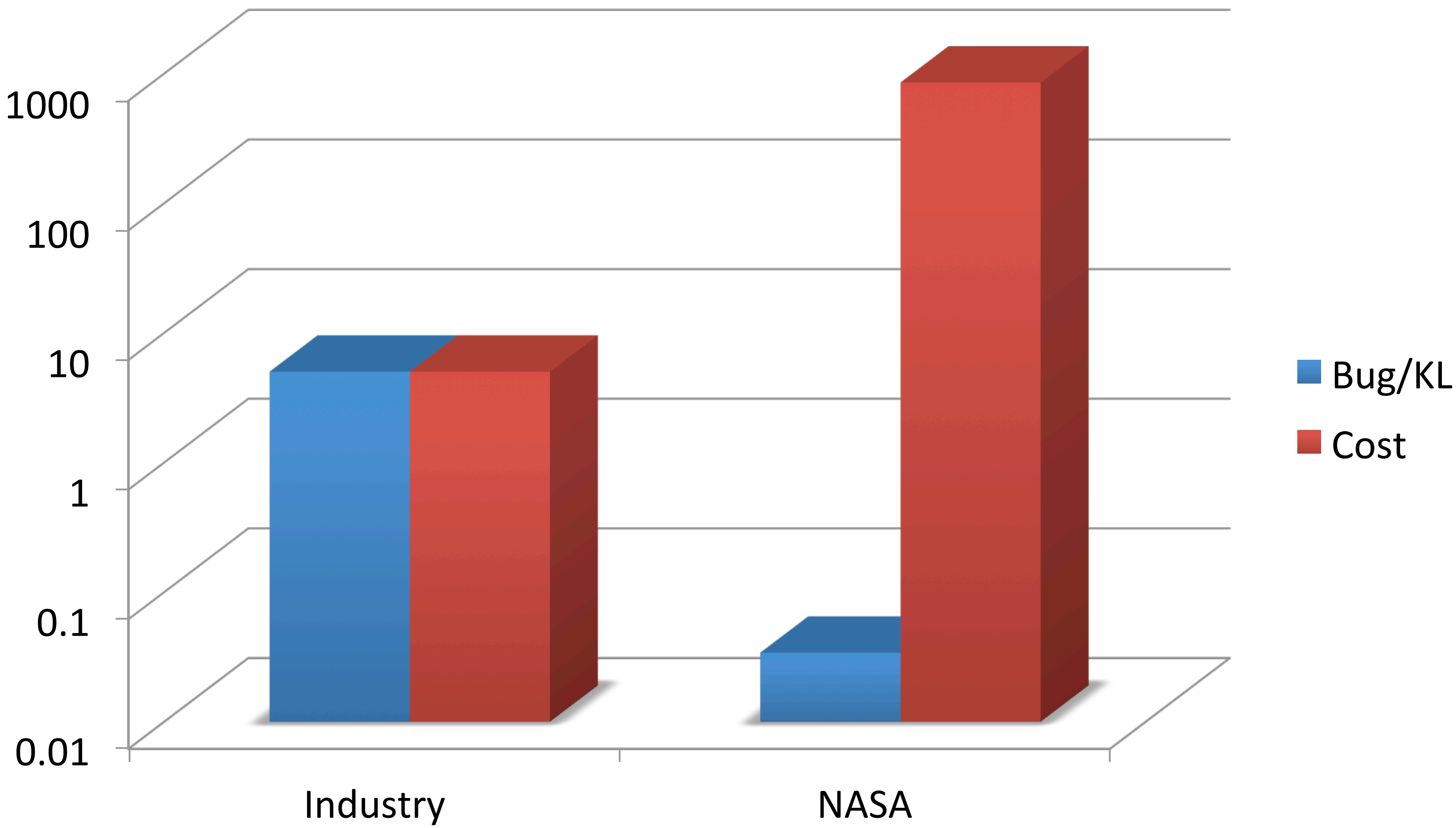
The Problem



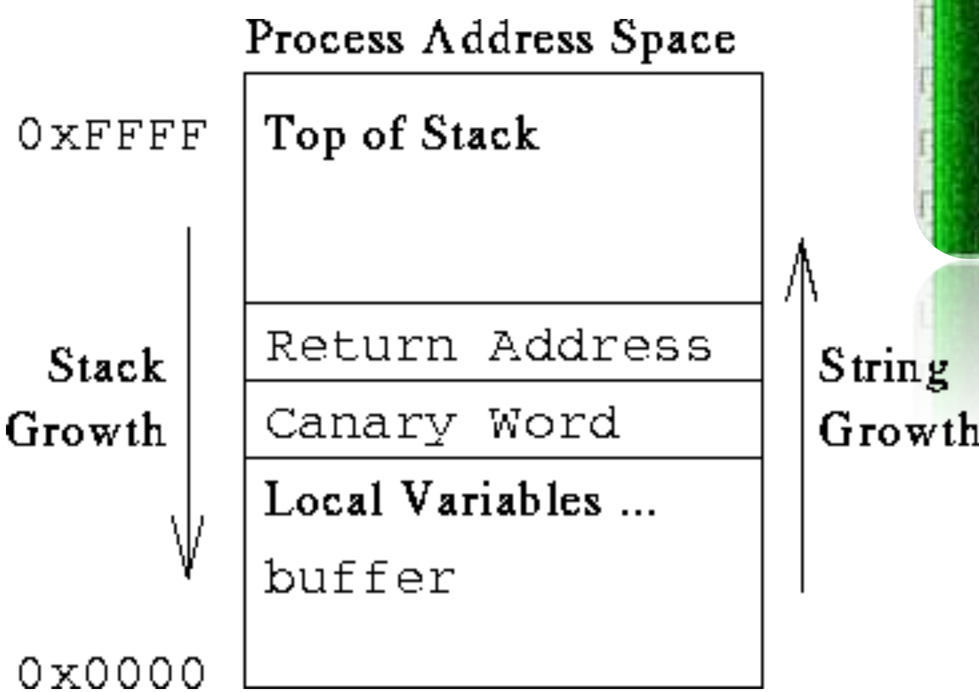
kryptus

The Problem

The Problem



Attacks

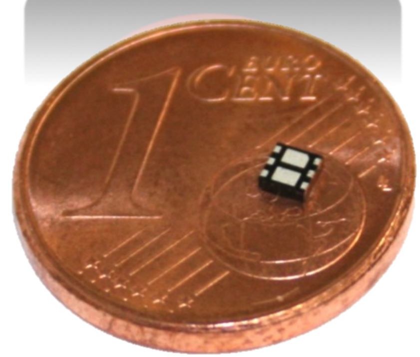

Process Address Space
0xFFFF
Top of Stack
Return Address
Canary Word
Local Variables ...
buffer
0x0000
Stack Growth
String Growth


www.AeroSoftCorp.com

Policies

- **Least privileges**

- **TCB minimization**



- **Security in Layers**

kryptus

Agenda

- **The Audience**
- **The Problem**
- **Our Proposals**
- **Implementation Results**
- **Current and Future Work**

kryptus

The Audience

- **Generally speaking, security professionals**
- **More precisely:**
  - People managing projects (PMP)
  - People analysing system security
  - People designing secure hardware
- **Problem Importance:**
  - The DHS (NSA, DoD, NIST) Cyber Security Roadmap ranks “Trustworthy Scalable Computing - TSC” the top 1 security challenge for the next years
  - Proper TSC requires trusted computing bases – TCB (software + hardware)

kryptus

The Problem - General View

- **Importance of real HW + SW system implementations is fully established**
- **However, even when ample resources are employed, this objective is very hard to attain**
- **Symptom: “is this system secure?” vs “for how long will this system remain secure?”**
- **The stochastic nature of real system implementations must not be ignored**
- **System examples: DRM enabled devices, HSM, DRE, Token, Game Console**

The Problem – Some Reasons

- **"Performance vs features vs TTM" race gains more attention than security does, from Industry**
- **Security is a highly interdisciplinary field and requires a unified view**
- **With current systems, complexity is intractable (even logically, not to mention the physical, probabilistic and human aspects)**
  - Formal proofs for HW + SW logical interactions shown to be NP-hard or intractable
- **There is NO Unified Security Theory**

kryptus

The Problem – Some Questions

- How to design and build secure systems?
- How to measure if a given architecture is better than others, prior to deployment?
- How to quickly assess the impact of a system modification?
- How to consider the lack of knowledge about some system characteristics?
- How to succeed even when no component is 100% trusted?
- How to consider at the same time physical and logical aspects?

kryptus

Our Objective

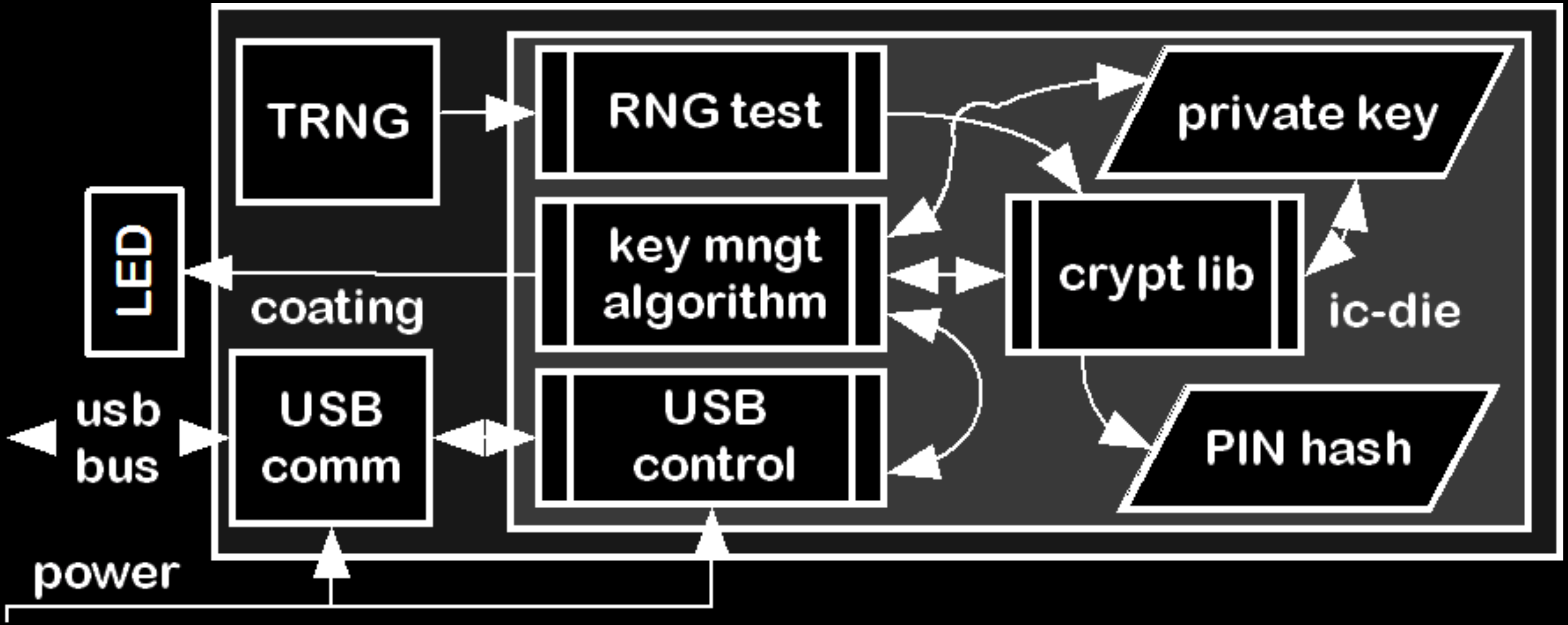
- **Reduce** design **hassle** of secure hybrid systems
- **Handle** the growing **complexity**
- **Automate** as much as possible the security **analysis** by using the design files as input
- Allow design of **trustworthy** systems with **faulty** components
- **Prevail** even when many **unknown aspects** are present

kryptus

Our Contributions – Overview

- The first framework that deals with early design stages of hardware-based secure systems with broad scope
- A tool that can be applied earliest in the design cycle of secure systems
- A probabilistic model, under which some well know “golden policies” can be proven and others be challenged

System Example – Crypto Token



System Observations

1. A secure system can be composed of other systems (or components);
2. The security of systems has a probabilistic nature;
3. Individually insecure (with respect to a given policy) components can be arranged in the form of a secure system;
4. Secure components (with respect to a given policy) may be arranged into an insecure system;
5. Ultimately, all components are physical. Logical components are abstractions represented in a particular physical component configuration (or state);
6. There are no complete descriptions of non-trivial practical systems;
7. Every component has an associated cost for its deployment;
8. Certain (typically local) components are associated with adversary rewards;

Extracted Properties I

B1. Interaction channel: every subsystem that can be composed with others has one interaction channel. This interaction channel may be a logical abstraction, providing a communication channel. The channel can be directed or not.

B2. Entropic potential: represents the information assets that generate benefits for the opponent. Measured in bits.

B3. Entropic impedance (or resistance to leakage): quantifies the permeability of components and interaction channels to entropy. It is given as the probability that a given entropy amount migrates in a given timeframe from A to B trough an channel AB.

Extracted Properties II

B4. Implicit security: components with a certain set of security policies are subject to different attacks. Each attack has a different cost and a different success probability.

B5. Security provided: expresses the ability an (directional) interaction has of transporting the implicit security experienced by a component A to a component B. Together with the implicit security, it expresses the ``protection relationship''.

Models

- **Our observations and properties are used to produce models where security characteristics can be explored**
- **We present in this paper three models:**
  - Two are graph-based:
    - Model 1: Bit leakage
    - Model 2: Adversary path (not shown here)
  - One is based on Decision Theoretic Probabilistic ProLog - DTProbLog

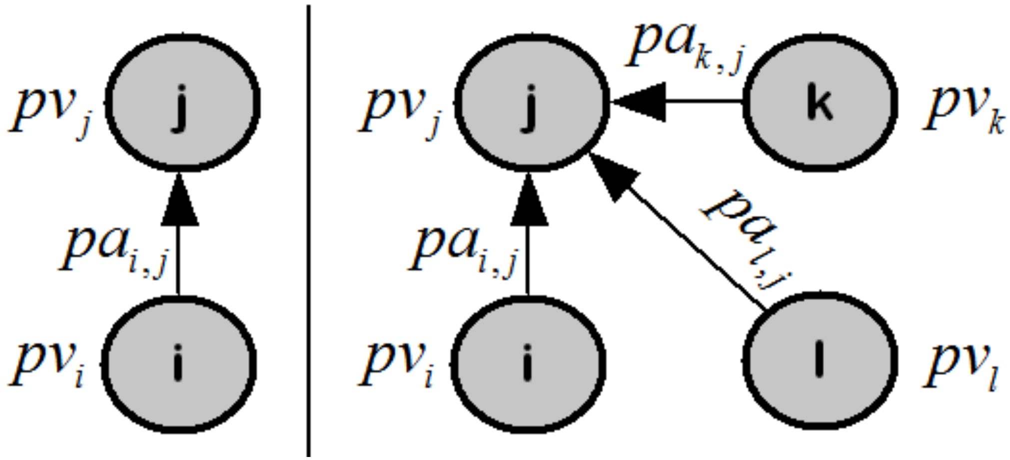
Graph Model 1 – Bit-Leakage

- **Uses Properties B1, B2, and B3: interaction channel, entropic potential, and entropic impedance**
- **Let *D = (V, A)* be a digraph representing a related system and external agents that interact with it**
- **Each vertex *i* from V represents a system component or a principal. Each arc *ij* from *A* represents a interaction channels (B1).**
- **Let *s* be a bit of the secret (B2) which the system protects and that the adversary aims**

Graph Model 1 – Bit-Leakage II

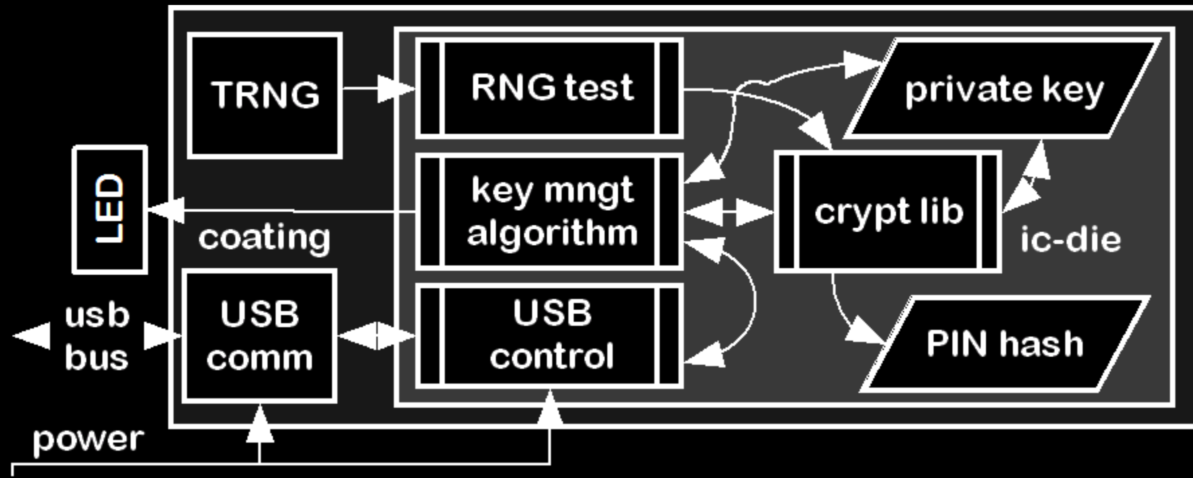
- **Vertex *i* has probability pv_i of knowing *s***
- **By properties B1 and B3, *s* leaks from its container (say i) through the arcs *ij* with probability $pa_{i,j}$**
- **We are interested in minimizing pv_k for the vertex *k* that represents the attacker**

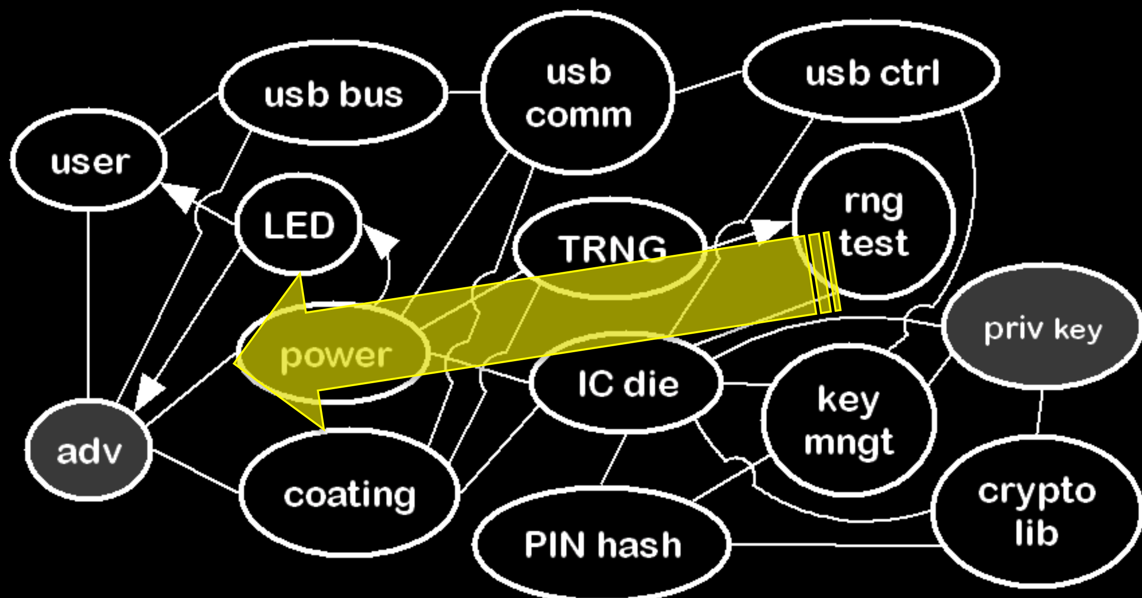
kryptus



$$pv_j = pa_{i,j} \times pv_i$$

$$pv_j = 1 - \prod_{i \in N_D^-(j)} (1 - pv_i \times pa_{i,j})$$


TRNG
RNG test
private key
LED
key mngt
algorithm
crypt lib
coating
ic-die
usb
bus
USB
comm
USB
control
PIN hash
power


usb bus
usb
comm
usb ctrl
user
LED
TRNG
rng
test
priv key
power
IC die
key
mngt
adv
coating
crypto
lib
PIN hash

Graph Model 2 – Attack Path

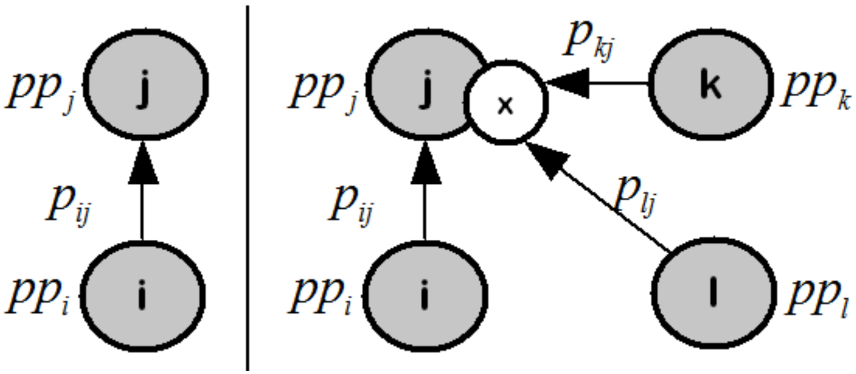
- **Uses Properties B1, B4, and B5: implicit security, security provided**
- **Let *D = (V, A)* be a connected digraph representing part of a system.**
- **Each vertex *i* of V represents a system component that can establish relations of protection. To each vertex *i* there is a related cost e_v. Each arc *ij* of A represents protection relationships.**

kryptus

Graph Model 2 – Attack Path

- **By B4, for each arc ij of A there is a violation cost c_{ij} associated with a given probability of succesful attack pp_{ij}.**
- **The arcs incident on j can be composed in and/or form.**
- **Let C be a subset of V representing the system's CSP. To each vertex j of C is associated a gain g_j.**
- **We are interested in making the best attack plan more expensive than the expected gain for the adversary.**

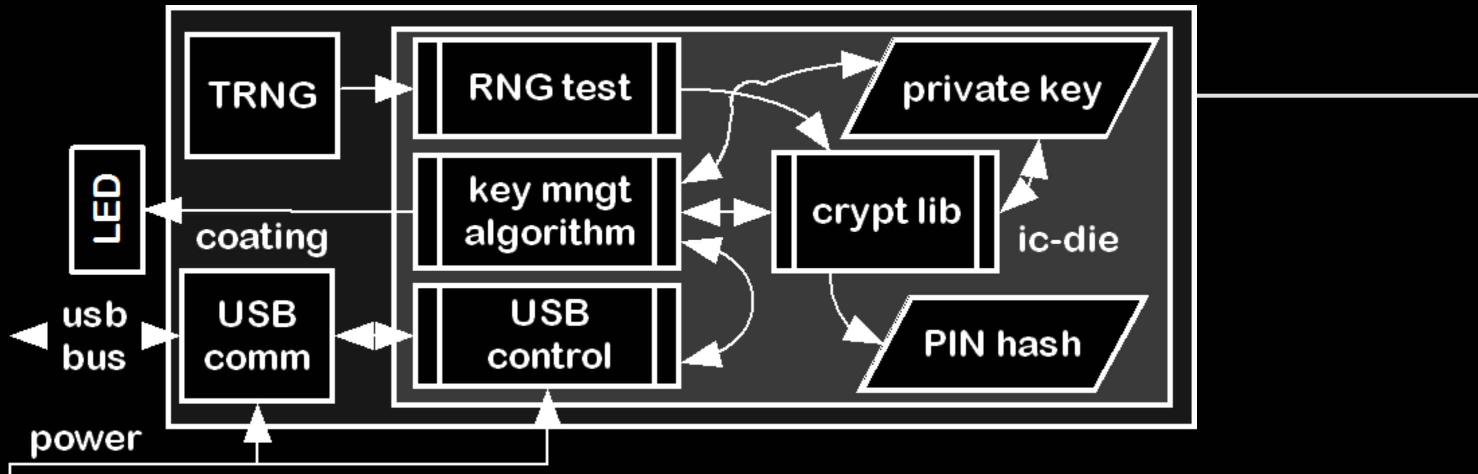
kryptus

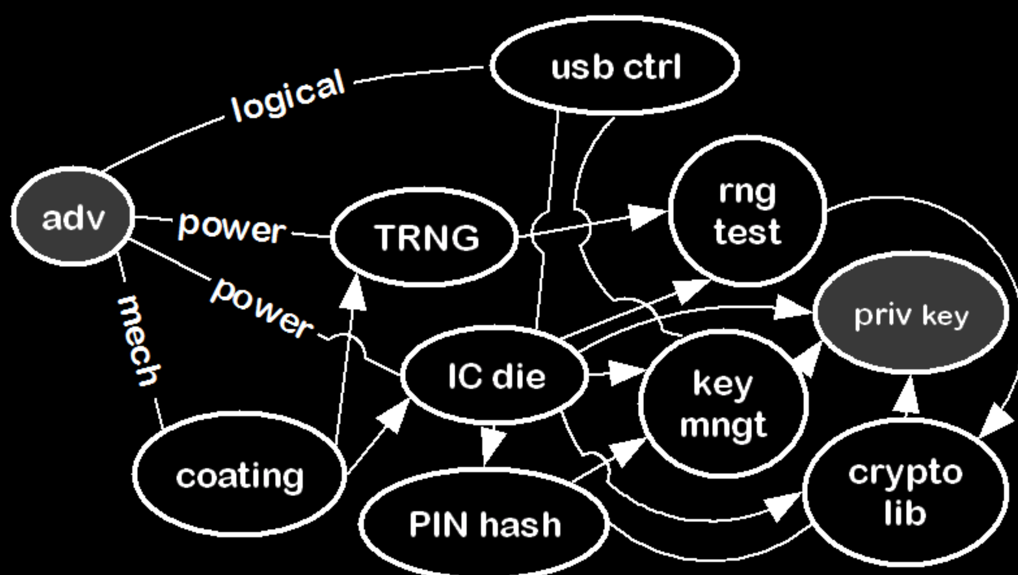


$$pp_j = 1 - \prod_{X \in N_D^-} \left(1 - \prod_{i \in X} p_{ij} \right)$$

$$e_j = e_{ij} = \frac{f(p_{ij})}{p_{ij}} + e_i$$

$$e_j = min \left(\sum_{x \in X} e_{xj}, \sum_{y \in Y} e_{yj}, ..., \sum_{z \in Z} e_{zj} \right)$$


TRNG
RNG test
private key
LED
key mngt
algorithm
crypt lib
coating
ic-die
usb
bus
USB
comm
USB
control
PIN hash
power


usb ctrl
logical
adv
power
TRNG
rng
test
power
mech
priv key
IC die
key
mngt
coating
crypto
lib
PIN hash

Model Results – Policy 1

- **Policy 1: "Grant system principals the least privileges necessary to perform their jobs"**
- **Theorem 1: Policy 1 either does not affect, or it improves the overall system security regarding confidentiality CSPs**
- **Proof 1: Comes from equation for pv_j in model 1 by arc removal where *j* is the vertex that represents the adversary**

kryptus

Model Results – Policy 2

- **Policy 2:** **“Minimize the size of the Trusted Computing Base”**
- **Theorem 2:** **Policy 2 does not always hold for integrity CSPs**
- **Proof 2:** **We use model 2. It suffices to show that we can arbitrarily increase system security by increasing the size of the TCB**

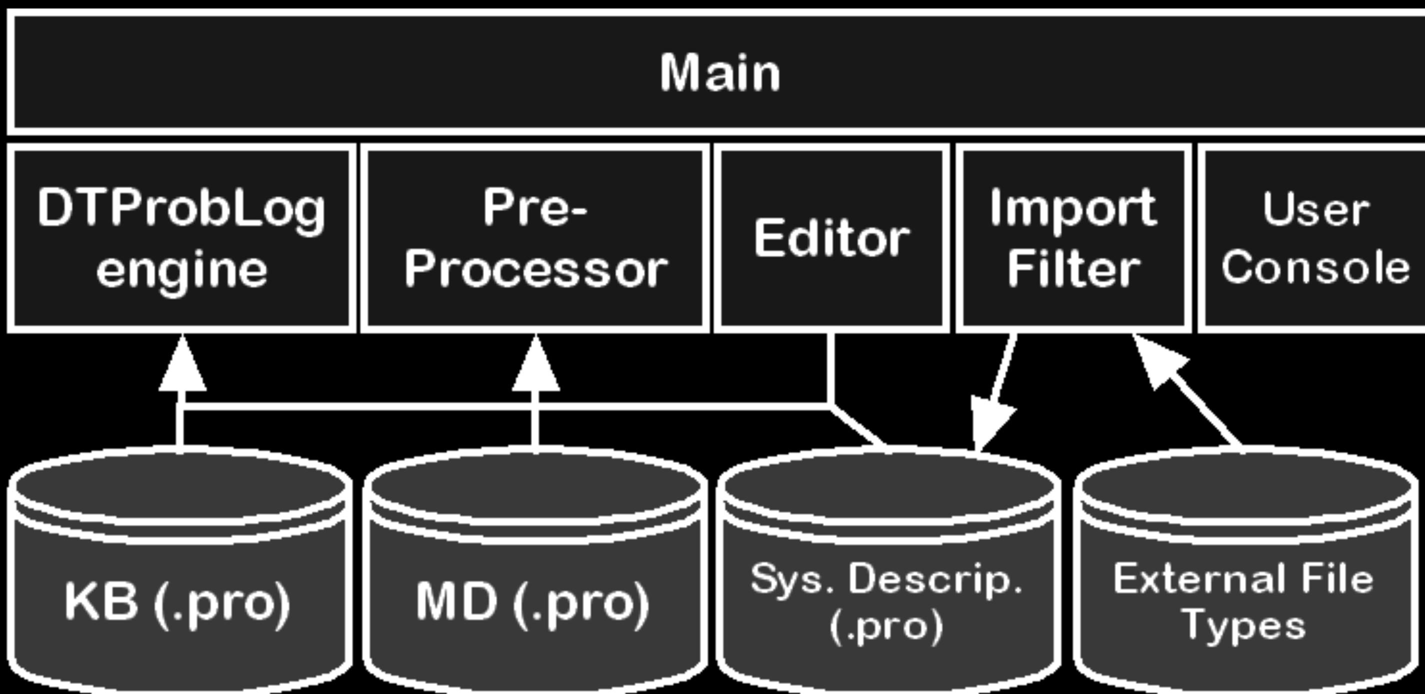
DTProbLog Model

- **Graph model limitations motivated the use of alternative models**
  - Too many annotations for richer descriptions
  - Limits representation for automation
  - Difficult to represent conditional probabilities
- **We chose Decision Theoretic Probabilistic ProLog language**
  - DTProbLog is a recent extension to ProbLog

DTProbLog – Some Features

- **Probabilistic facts and queries in KB:**
  - e.g. 0.9: protects_directed (J, I).
- **Optimization target (or decisions):**
  - e.g. ? :: attacked(C) :- component(C).
- **Utility functions (or costs and gains):**
  - e.g. break_policy(C) => - 5 :- component(C).
- **Because ProLog is expressive, it allowed us to describe all B1..B5 properties**

FORTUNA Tool Architecture



kryptus

Main Data Components

- **Model Rules (MB):**
  - Encodes variations of models 1 and 2
  - Supporting tools (e.g. traversing rules, SP calculation, cost adding…)
- **Knowledge Base (KB):**
  - Encodes best current values for costs, bug density, probabilities
  - Initially with industry's defects/kloc metric
- **System Description (SD):**
  - Contains the system description
  - Either from user input or "Import Filter"

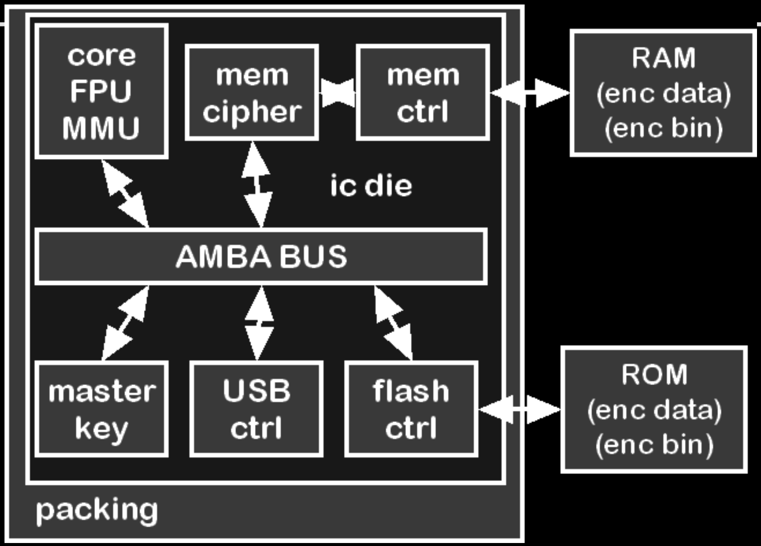
Example Query

```
?- dtproblog_solve(Strategy,ExpectedGain).
ExpectedGain = 17.12121,
Strategy = [attacked(coating),
     attacked(die),attacked(priv_key)]
```

Tool Case Usage

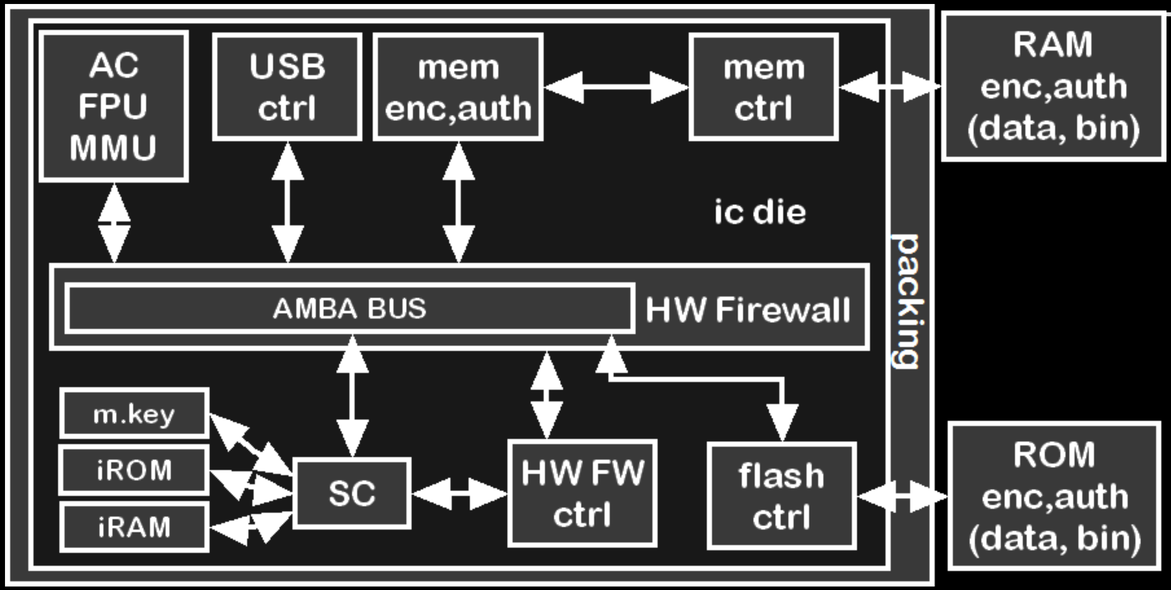
- **We successfully employed FORTUNA during the development of a Cryptographic Secure Processor (SCUP)**
- **FORTUNA allowed 4x faster security design reviews, automatic analysis and cost reduction.**
- **Tool guided important architectural improvements**

SCUP – Early Design




```
?- problog_max(path(adversary,master_key),
      Prob, Strategy).
Strategy = [attacked(usb_ctrl),
      attacked(usb_stack),
      attacked(running_bin),
      attacked(master_key)]
Prob = 0.04809
```

SCUP – Final Design



- **Evolved to a Multi-core Asymmetric Processor**
- **Motivated a second core with minimal software stack**
- **Motivated the HW Firewall**

kryptus

Conclusion

- FORTUNA brings both practical and theoretical contributions for hardware-based systems' design
- Models could be used to prove (or challenge) some golden rule heuristics
- The tool was used in the design of a Cryptographic Secure Processor, easing the development process

kryptus

Current and Future Work

- **Improve KB precision through usage data feedback**
- **Improve MD with new models from properties B1…B5**
- **Develop new CAD plugins to make the target system description even faster**
- **Adjust model's equations to directly support conditional probability (correlation)**

Thank You!

Questions?

gallo@kryptus.com

gallo@ic.unicamp.br