Mathematical Methods, Models, and Architectures
for Computer Networks Security (MMM-ACNS-2010)

CLARIFYING INTEGRITY CONTROL AT THE TRUSTED INFORMATION ENVIRONMENT

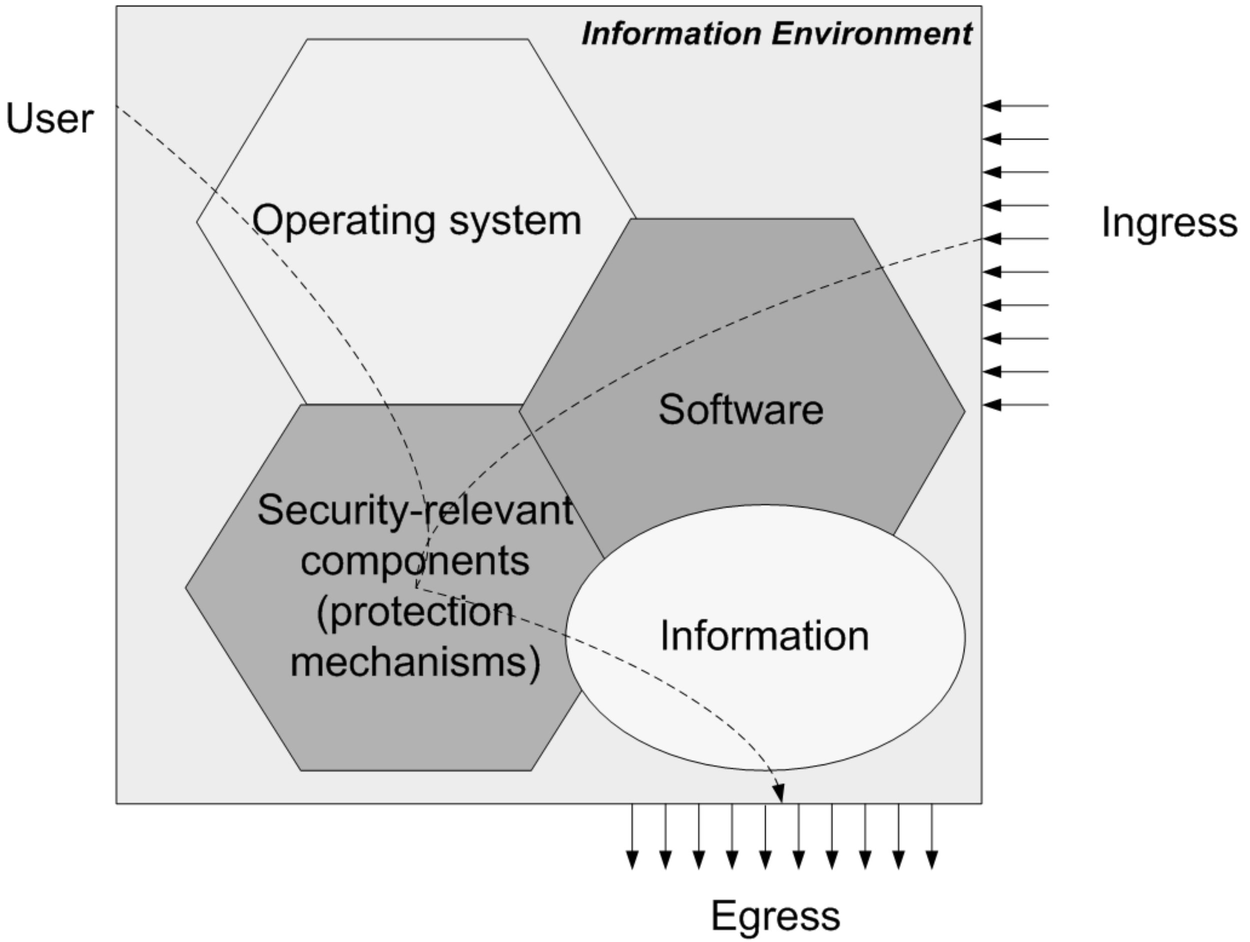
Dmitry Zegzhda Peter Zegzhda Maxim Kalinin

Information Security Center,
St. Petersburg Polytechnical University,
Russia

September 8-10, 2010

Trusted computer system

- Confidentiality
- Integrity
- Accessibility



How to reach the trustworthiness?

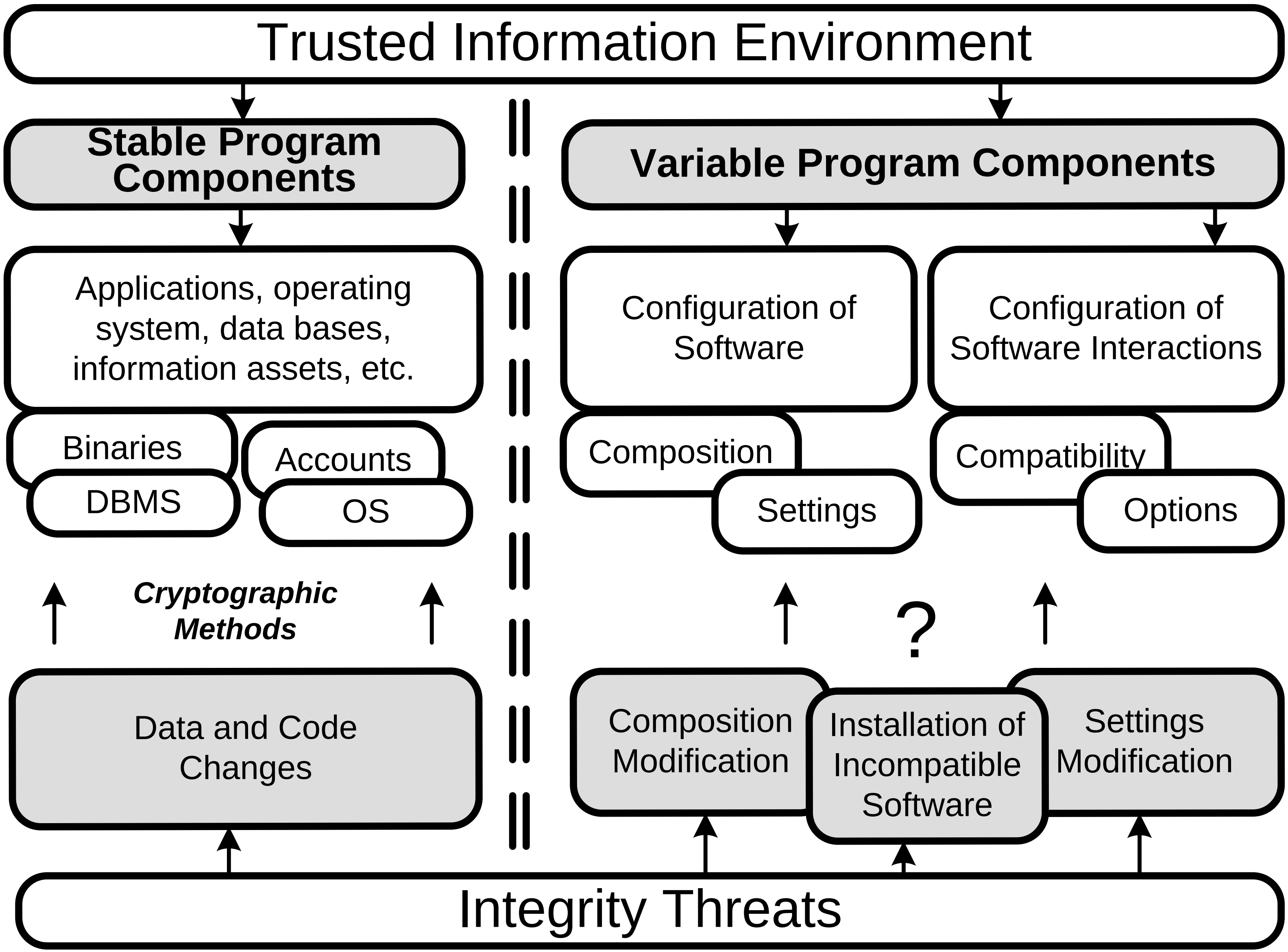
- **Source code analysis**
  - Reliability models
- **Security modeling and assurance**
  - Discretionary, mandatory, role-based, etc. models
  - Security specification languages, calculus and processing tools
  - Security monitoring and vulnerabilities detection
  - Intrusion detection methods
- **Cryptography**
  - Cryptographic algorithms and protocols

- **Result: ‘point’ security.**
  - ✓ BUT INFORMATION ENVIRONMENT CONSTANTLY CHANGES

Integrity problem

- **Confidentiality**
- **Accessibility**
  - Traditional methods
- **Integrity**
  - Data-relevant definition: assurance that information is authentic and complete (hash, checksums)
  - Functional integrity (wholeness of the system)?
    - ✓ Contradictory versions of the program libraries
    - ✓ Software Updates
    - ✓ New access permissions for new users

Components of information environment



Stable vs. variable

- **Stable components:**
  - the functional modules that are founded at system designing and building (executables, OS elements, data bases)

  Long life-cycle -> cryptographic methods

- **Variable components:**
  - modified settings (security parameters: system registry, access control rights; session characteristics: active users, applications list
  - Huge number of parameters undergoing control

  Short and tiny life cycle -> ?

***Integrity* is ensuring**
that information environment is stable (invariable)
(not in point but in area)

System state elements

a set of program components $p_i \in P$, where P depicts the set of TIE's components, $i \in N$. A program item is specified with a program type $T_n \in T$, where T is a set of program types (e.g., system software, user application, security mechanism), $n \in \mathbf{N}$;

a set of program attributes $A^{T_n} = \{a_j^{T_n}\}$, where T_n is a program type, a_j is a component of program attribute; $j \in \mathbf{N}$. Program attributes are the settings of the TIE's program components;

a set of attribute values $V^{T_n,p_i} = \{v_k^{T_n,p_i}\}$, where $\forall v_k^{T_n,p_i} = var(p_i, T_n, A^{T_n}), k \in \mathbf{N}$. Function $var: P \times T \times A^T \to V^T$ for the program item $p_i \in P$ of type $T_n \in T$ with attributes A^{T_n} returns the values V^{T_n}.

Formal integrity conditions

$ref : P \times T \times A^T \times V^T \rightarrow P \times T \times A^T \times V^T$:

set of attributes $a^t \in A^T$ with values $v^{t,p} \in V^{t,p}$

program component $p \in P$ of the type $t \in T$

points to the set of AGREED attributes $a^{t'} \in A^T$ with values $v^{t',p'} \in V^{t',p'}$ of another program item $p' \in P$ of the type $t' \in T$.

One (V^T) or several ($V^T \pm \Delta V^T_{\Leftarrow}$) values refer to another program item: $ref : P \times T \times A \times V^T \rightarrow P \times T \times A \times (V^T \pm \Delta V^T_{\Leftarrow})$.

The reverse function $ref^{-1} : P \times T \times A \times (V^T \pm \Delta V^T_{\Leftarrow}) \rightarrow P \times T \times A \times (V^T \pm \Delta V^T_{\Rightarrow})$ defines area $V^T \pm \Delta V^T_{\Rightarrow}$ for each point from $V^T \pm \Delta V^T_{\Leftarrow}$.

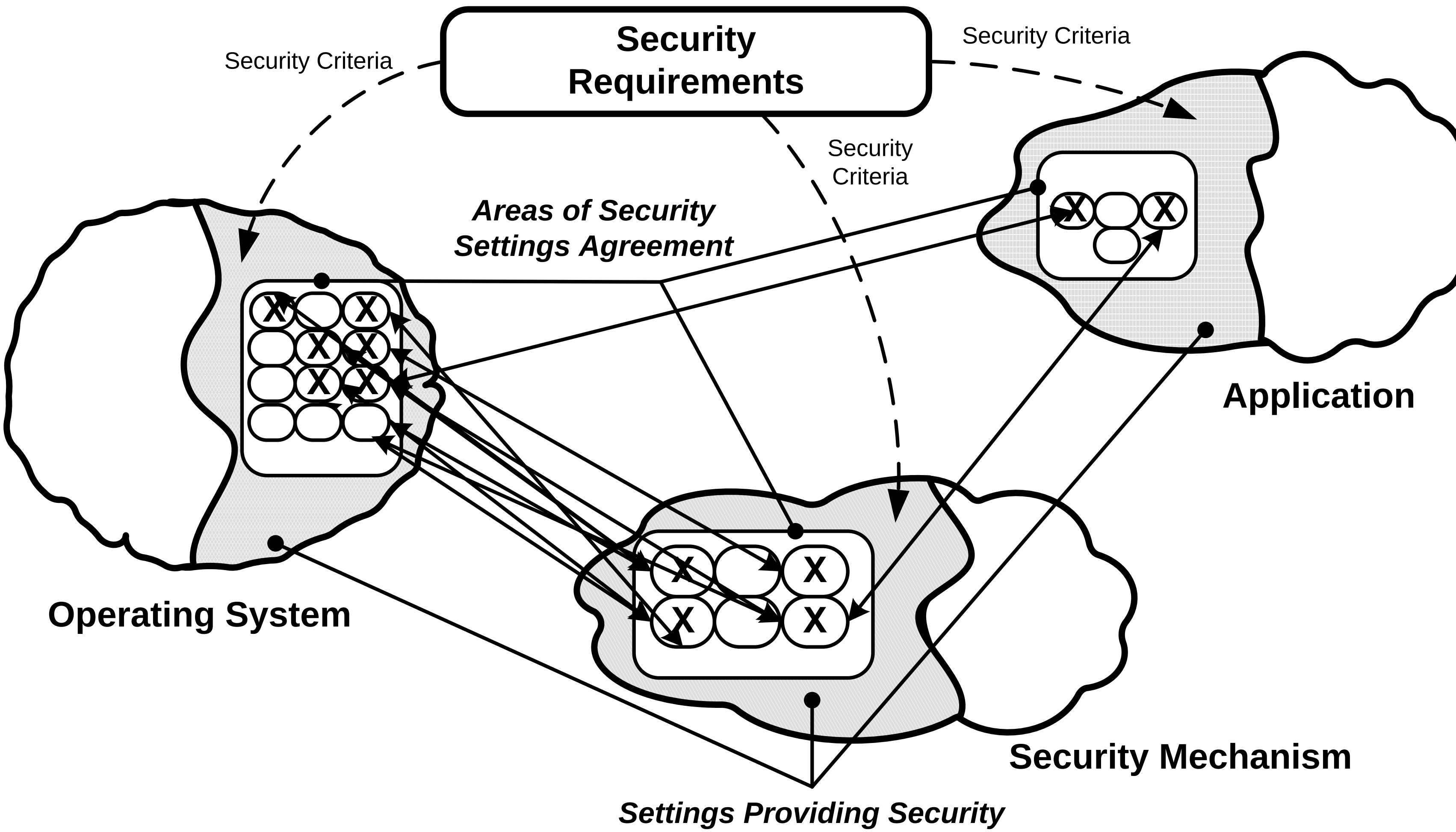
Symmetric relations has not to be empty:

$$\forall p \in P, \forall t \in T \quad \exists a \in A^t : \exists p' \in P, \exists t' \in T, \exists d_{\Rightarrow} = V^{p,T} \pm \Delta V^{p,T}_{\Rightarrow} \ u$$

$$d_{\Leftarrow} = V^{p,T} \pm \Delta V^{p,T}_{\Leftarrow} : \ ref(p,t,a^t,d_{\Rightarrow}) = < p', t', a^{t'}, d_{\Leftarrow} >;$$

$$ref^{-1}(p', t', a^{t'}, d_{\Leftarrow}) = < p, t, a^t, d' >; d' \cap d_{\Rightarrow} \neq \varnothing.$$

Graphical interpretation



Implementation: security control system