

TrustedFlow: a Method for Remotely Authenticated Operations

Yoram Ofek

Department of Information and Telecommunication
University of Trento

Joint work with:

- Prof. Mario Baldi – Politecnico di Torino
- Dr. Moti Yung – Columbia University

RPI -NSF Workshop: 29-30 April 2004

1

The Context of TrustedFlow

- **A method that combines:**
 - **Computing and networking**
 - **For distribution of trust
or entrusting**
- [Stand-alone computer is well-protected, by definition]

2

The Problem

- *How to ensure run-time SW authenticity*
- *Focusing on two generic protocols:*
 - **1. Sending data:**
 - *To avoid unfair-usage/attacks on networks/servers*
 - *TCP, SLA, 802.11, GRID ...*
 - **2. Receiving data (e.g., content):**
 - *To ensure digital right management (DRM)*
 - *Audio, video ...*

3

Some Current Approaches

- **1. Sending data**
Based on monitoring & diagnostics
reactive (after the fact)
 - Firewall monitoring
 - SLA policing
- **2. Receiving data (e.g., content)**
Based on adding special HW
 - E.g., TCPA/Palladium, "watch-dogs," ...
 - Specific to each computer HW depends on configuration, architecture, ...
 - What to do with current systems?

4

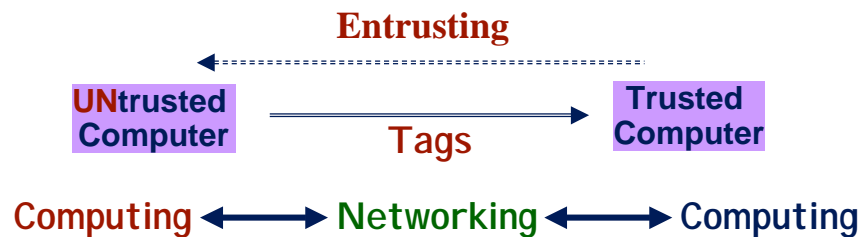
What Do We Want to Achieve?

Remote authentication of code during execution

Trusted 1st computer ensure that SW executed on
Untrusted 2nd computer was not modified
Thereby,
entrusting the 2nd computer

5

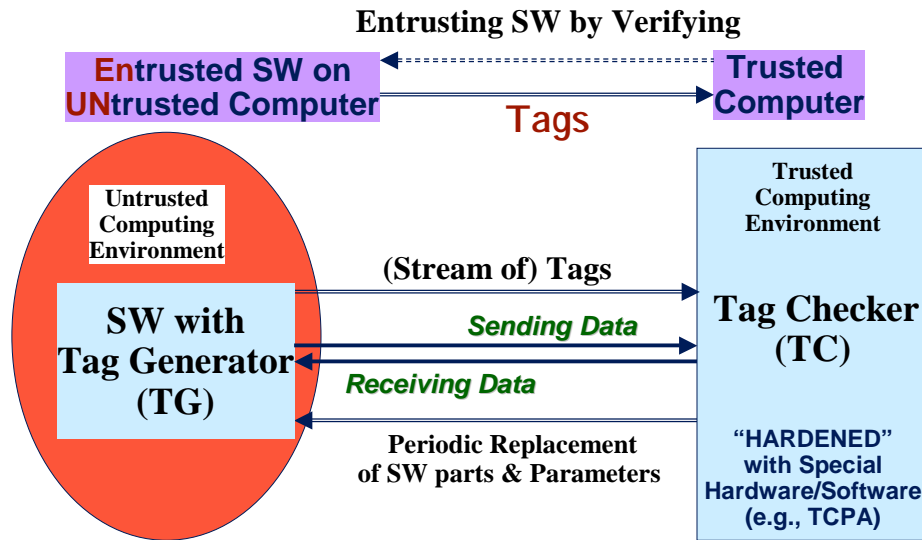
How: Entrusting



- (Stream of) Tags are EMANATED from a program=code=software at run-time
- ENTRUSTING by verifying the Tags

6

How: Entrusting



7

TrustedFlow Key Components

Tag Generator (TG) executes:
the combined module
to create and emanate tags

Tag Checker (TC) validates:
the tags

8

Principles for Constructing TG

- **Interlocking (mixing)**
 - Ensure the execution of the original SW
 - Such that, the trusted tag generation is subordinated to SW originality
 - **Correct tags imply original SW was executed**
- **Hiding**
 - Selected operations of the TG is secret
 - Hard to extract from the SW
 - (in a defined time interval)**
 - While ensuring proper interlocking

9

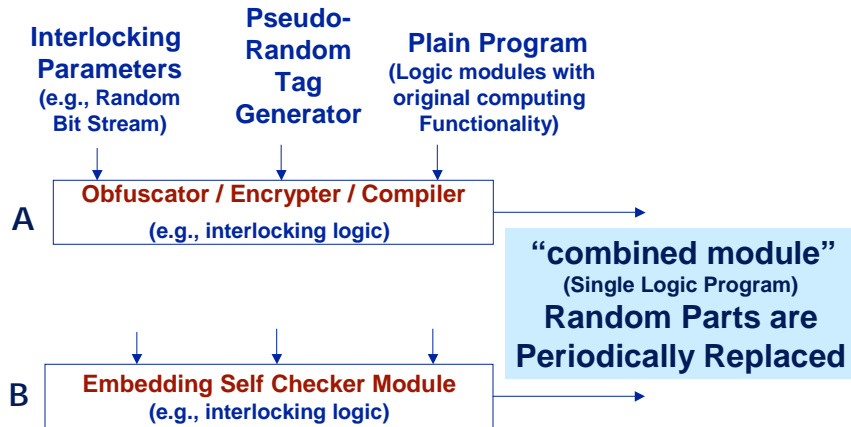
Realization Examples

- Interlocking
 - Mixing a pseudo random generator function in the original software
- Hiding
 - Obfuscation with
 - Periodic replacement (of random parts)

Software Engineering Challenges

10

Hiding – Combined Module



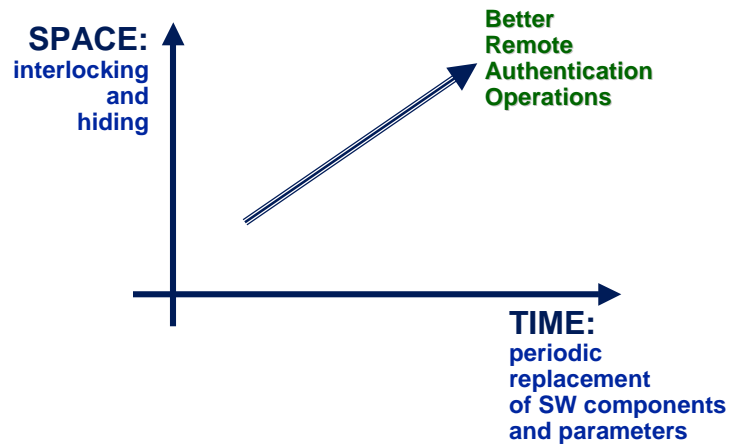
11

What is Obfuscation (background)

- An obfuscator **O** is a probabilistic "compiler" that takes as input a program **P** and produces **a new program O(P)** that has the same functionality as P yet is "unintelligible" in some sense. Most of obfuscator applications are based on an interpretation of the "unintelligibility" condition in obfuscation as meaning that **O(P) is a "virtual black box"**
- Code obfuscation is very similar to code optimization, but with obfuscation we are maximizing obscurity while minimizing execution time, whereas with optimization we are just minimizing execution time

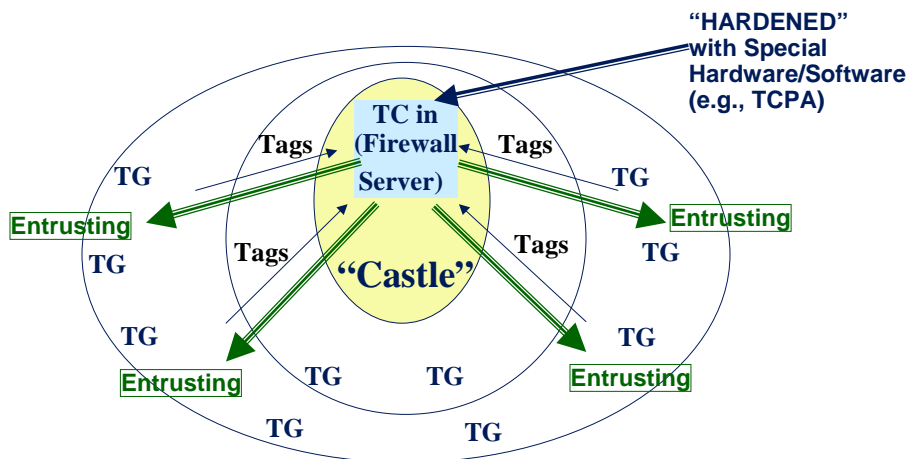
12

Quality of Remote SW Authentication



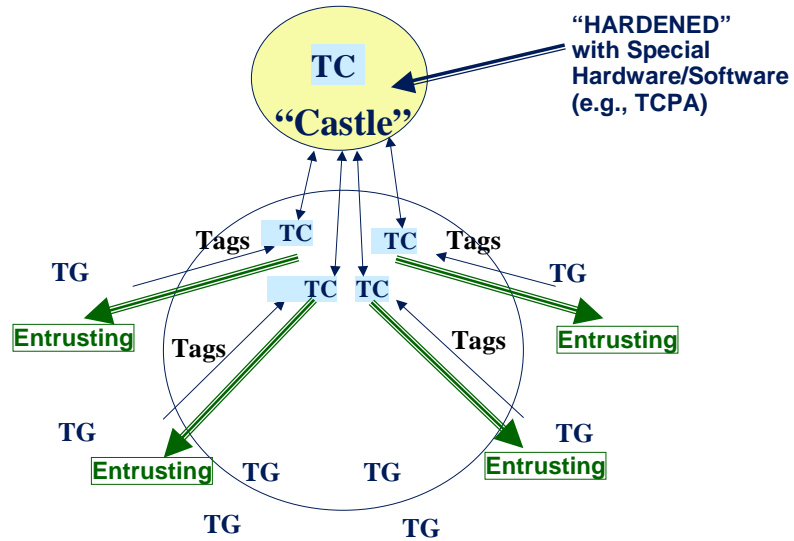
13

Distribution of Trust



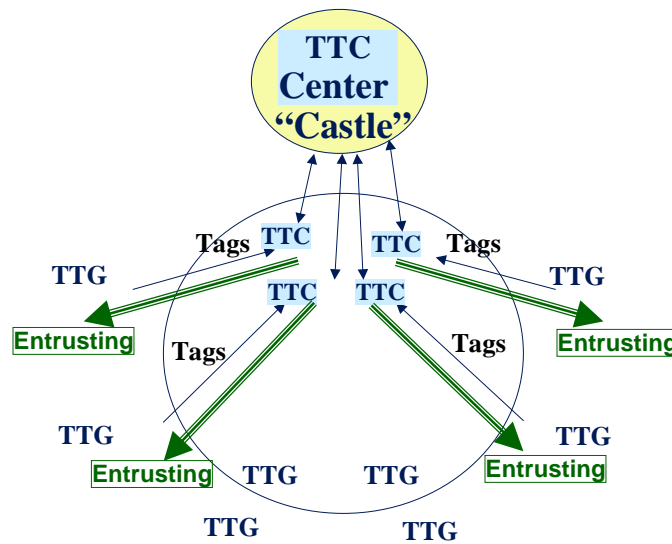
14

Hierarchical Distribution of Trust



15

Hierarchical Distribution of Trust

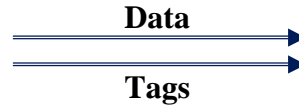


16

Modes of Operations

➤ **Direct data transfers:**

- For TCP and SLA
- For DoS/DDoS avoidance
- For authenticated access
- For mobile users
- For GRID computing



➤ **Reverse data transfers:**

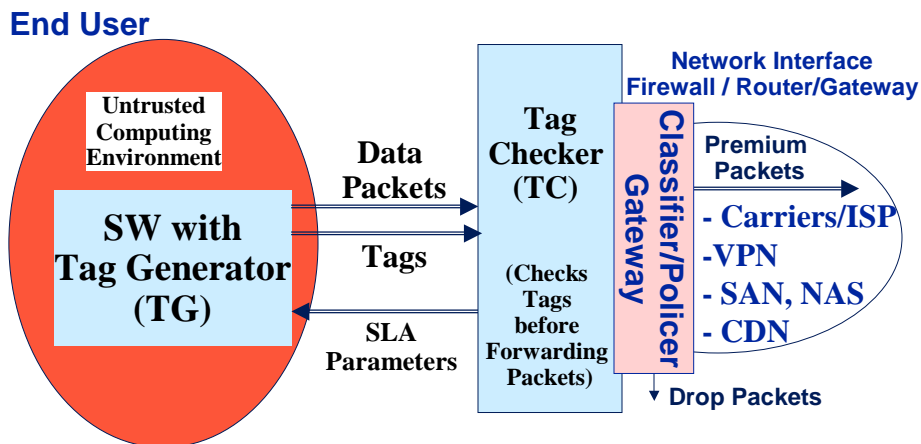
- Conditional playing
 - For remote enforcement of right management
- Remote run-time authentication of "watchdog" SW for trusted computing



17

Direct Data Transfers

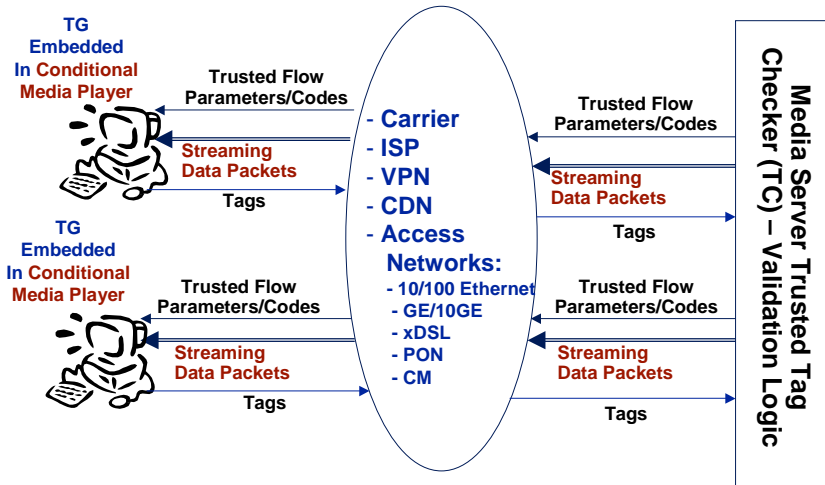
SLA / TCP Enforcement for Carriers/ISPs



18

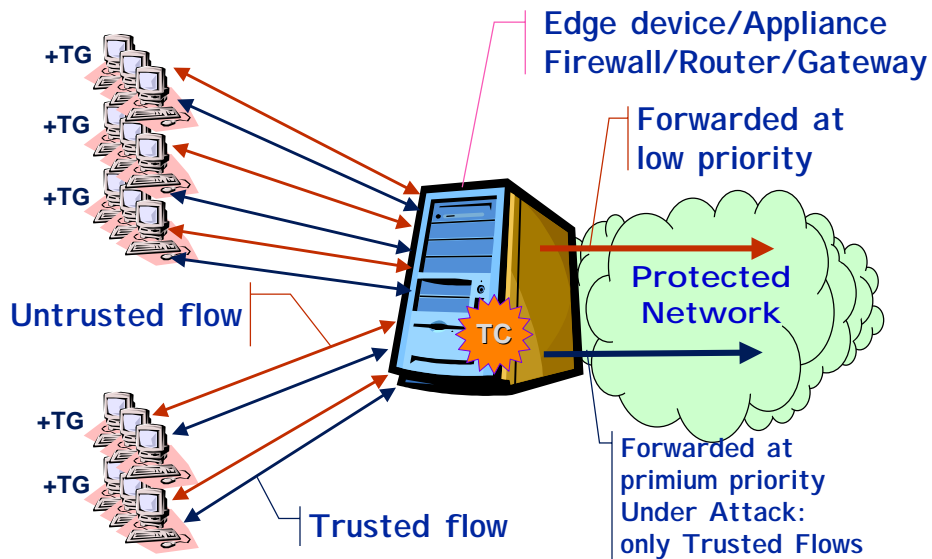
Reverse Data Transfers

Conditional playing enforcement of right management



19

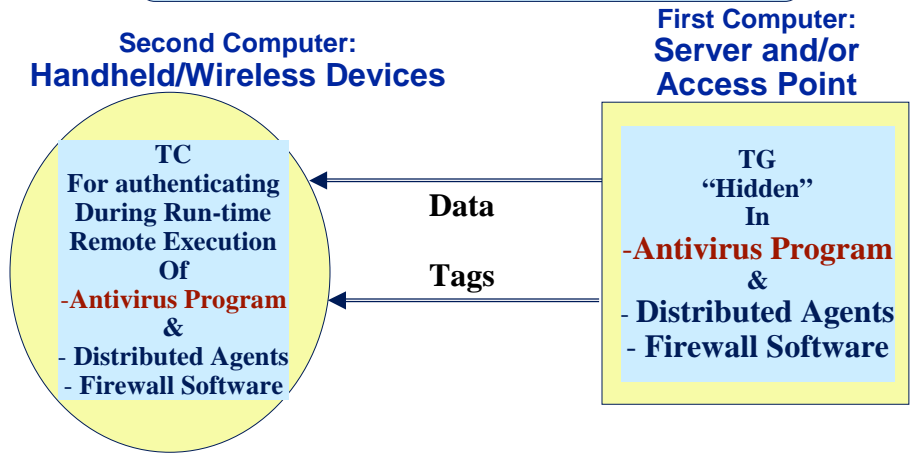
DoS Avoidance - Direct Transfers with Mapping to Premium Service



20

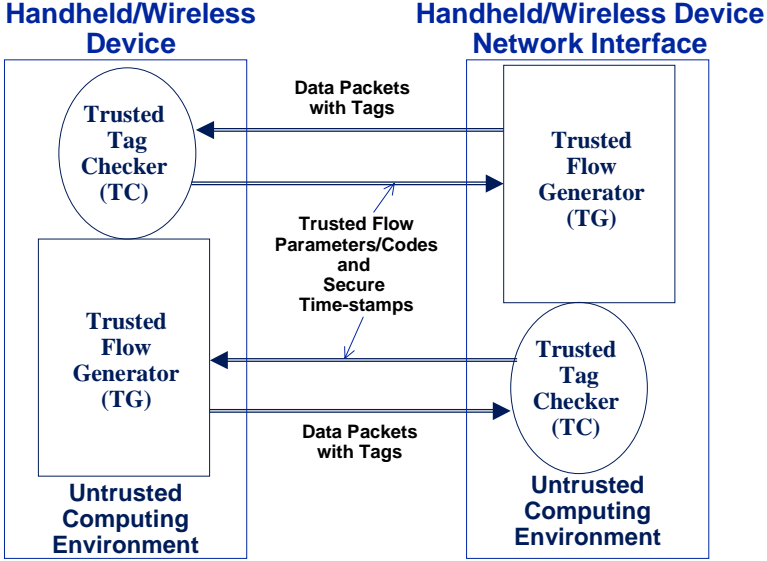
Direct Data Transfers Mobile/Ad-hoc TrustedFlow

For protecting handheld/wireless devices with limited computing/storage power



21

Mutual Trust



22

Synergy

- The TrustedFlow (TFG /TTG) tool offers advantages either by itself or together with other existing technologies in providing more secure solutions
- Complementary technologies
 - Crypto protocols: IPsec, SSL, PKI
 - Antivirus protection tools
 - Software integrity mechanisms: signed applets, software obfuscation
 - Peer-to-peer collaborative tools and distributed middleware

23

Realization Activities

- University of Trento (Prof. Fabio Massacci)
 - Continuous authentication prototype (works)
 - **Avoidance of: Hijacking, Intrusion, DoS, Trojan Horses**
 - Microsoft Research provided source code for Windows XP
- Politecnico di Torino (Prof. Mario Baldi)
 - Funding from Microsoft Research
 - Istituto Boella:
 - cryptographic solutions for tag generation
 - Software Engineering Group: code mobility
 - Software Reliability Group: self checking
 - Network Group: protocol specification

24

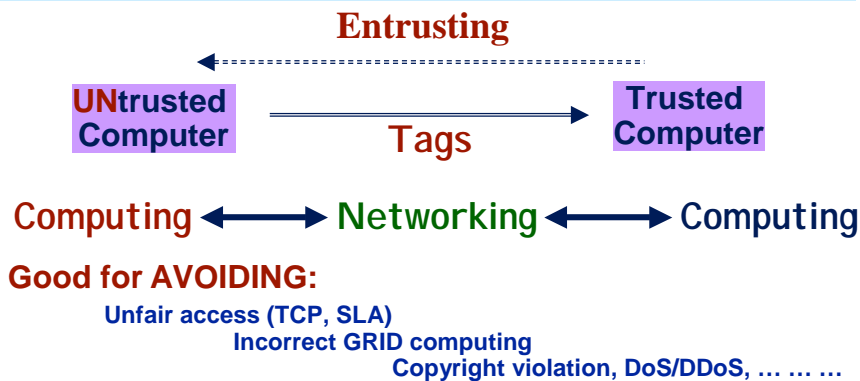
Add-on Software with Scalability: Minimum Run-time Overhead

- Trusted Tag Generator (TTG): add-on software to client
 - Generating 1-bit per packet during run-time (is sufficient)
 - Part of existing protocols: IP, TCP, Antivirus, E-mail, HTTP, Ping ...
- Trusted Tag Checker (TTC): add-on software to network interface
 - Checking 1-bit per packet during run-time
- Significantly lower complexity than current firewalls or other network interfaces:
 - Consequently:
 - Less expansive with high scalability - no special hardware
 - While ensuring uninterrupted service

25

Summary: Creating a Trusted Distributed Computing and Networking

- Relying on trusted computing entity to guarantee trusted execution of remote software
- Scalable with uninterrupted service (under attack)



26