An assessment of MAFTIA's Intrusion Tolerance Capabilities

presented by Robert Stroud,

University of Newcastle upon Tyne

Intrusion Tolerance Analysis of MAFTIA Architecture

- Idea is to show how the MAFTIA architecture can be deployed to protect a realistic application against a plausible set of attacks and vulnerabilities
- Of necessity, this is a paper study and incomplete
- Scenario is an e-commerce broker providing a virtual market place
- By comparing the resilience to attack of a MAFTIA version with a non-MAFTIA version, we can demonstrate the “added value” of MAFTIA’s mechanisms

Fault-forecasting process

- Evaluate system behaviour with respect to fault occurrence or activation
- Identify event combinations that lead to undesired events
- Our analytical approach is to use fault trees that represent possible intrusion scenarios,
- These explore the possible combination of events that could lead to a root event (such as stealing money from a company)
- Qualitative assessment of likelihood of primitive events

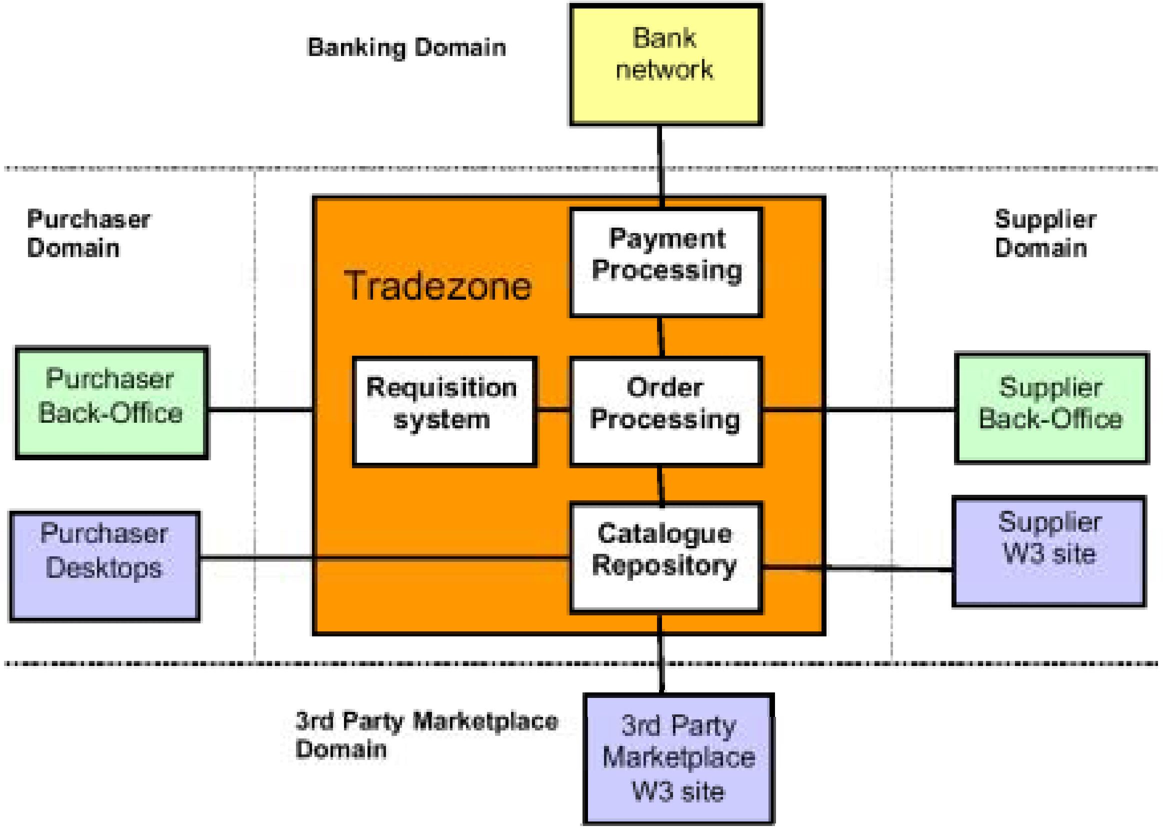
Approach

- We first developed a “use case” for the MAFTIA architecture, based on a simplified but realistic e-commerce application
- We then used fault trees to analyse a representative, but by no means complete, set of attack scenarios
- The fault trees illustrate the series of MAFTIA mechanisms that an attacker must successively overcome in order to achieve their objective
- The difficulty of achieving each step in the process is discussed as part of the fault tree analysis.
- This analysis highlights the ways in which MAFTIA’s architectural mechanisms support the construction of intrusion tolerant applications

Examples of MAFTIA's intrusion tolerance capabilities

- Group communication
  - Byzantine agreement protocols
  - Partially synchronous (TTCB)
  - Asynchronous (linear secret sharing)
- Cryptographic techniques
  - Threshold signature schemes
- Distributed authorisation service
  - Replicated authorisation servers
  - Local reference monitors
  - Java cards
- Intrusion detection systems
  - Secure channels
  - Replicated sensors
  - Diverse event analysers

Tradezone


Banking Domain
Bank network
Purchaser Domain
Supplier Domain
Tradezone
Payment Processing
Purchaser Back-Office
Requisition system
Order Processing
Supplier Back-Office
Purchaser Desktops
Catalogue Repository
Supplier W3 site
3rd Party Marketplace Domain
3rd Party Marketplace W3 site

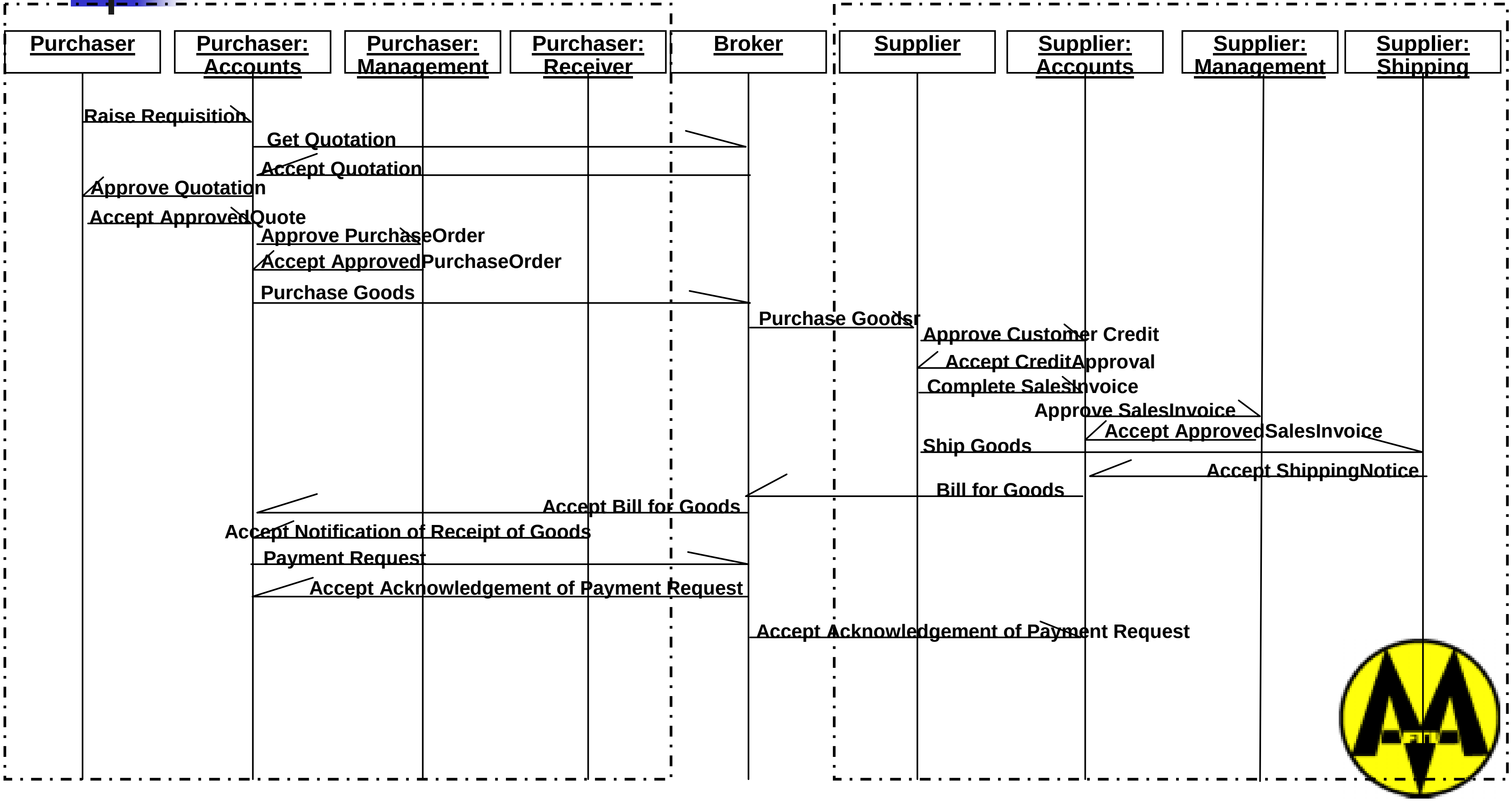
Security goals

- The high-level security goals of the Tradezone security policy are to provide a secure, and timely transaction service:
  - Purchasers should be correctly charged for goods they receive.
  - Suppliers should receive correct payment for goods supplied.
  - Suppliers should dispatch goods in a timely fashion once an order has been accepted.
  - Purchasers should confirm receipt of goods in a timely fashion.
  - Banks should process payment requests and post confirmation signals in a timely fashion.
  - The integrity and availability of supplier catalogues should be assured.
  - All transactions should remain confidential.

Security rules

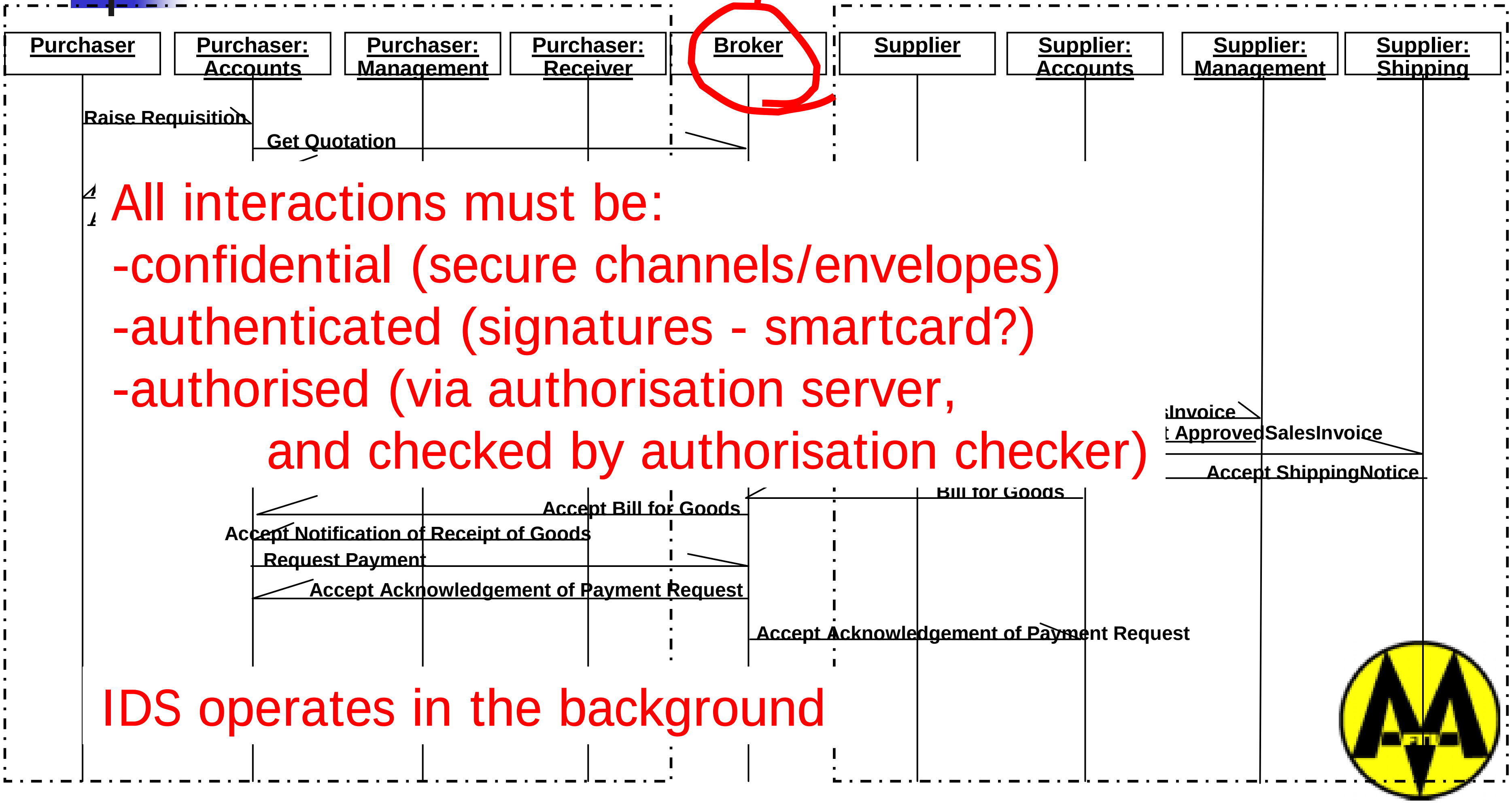
- In order to achieve these security goals, the Tradezone application will be designed to enforce security rules such as the following:
  - A registered purchaser should have read access to all the catalogues
  - A registered supplier should only have write access to their own catalogue
  - All communications should be authenticated and logged by Tradezone
  - All messages must be authenticated and encrypted and should conform to the purchasing workflow

Purchasing workflow



Purchasing workflow

Trusted 3rd Party (TTCB)



Purchaser

Purchaser: Accounts

Purchaser: Management

Purchaser: Receiver

Broker

Supplier

Supplier: Accounts

Supplier: Management

Supplier: Shipping

Raise Requisition

Get Quotation

Invoice

ApprovedSalesInvoice

Accept ShippingNotice

Bill for Goods

Accept Bill for Goods

Accept Notification of Receipt of Goods

Request Payment

Accept Acknowledgement of Payment Request

Accept Acknowledgement of Payment Request

All interactions must be:

-confidential (secure channels/envelopes)

-authenticated (signatures - smartcard?)

-authorised (via authorisation server,

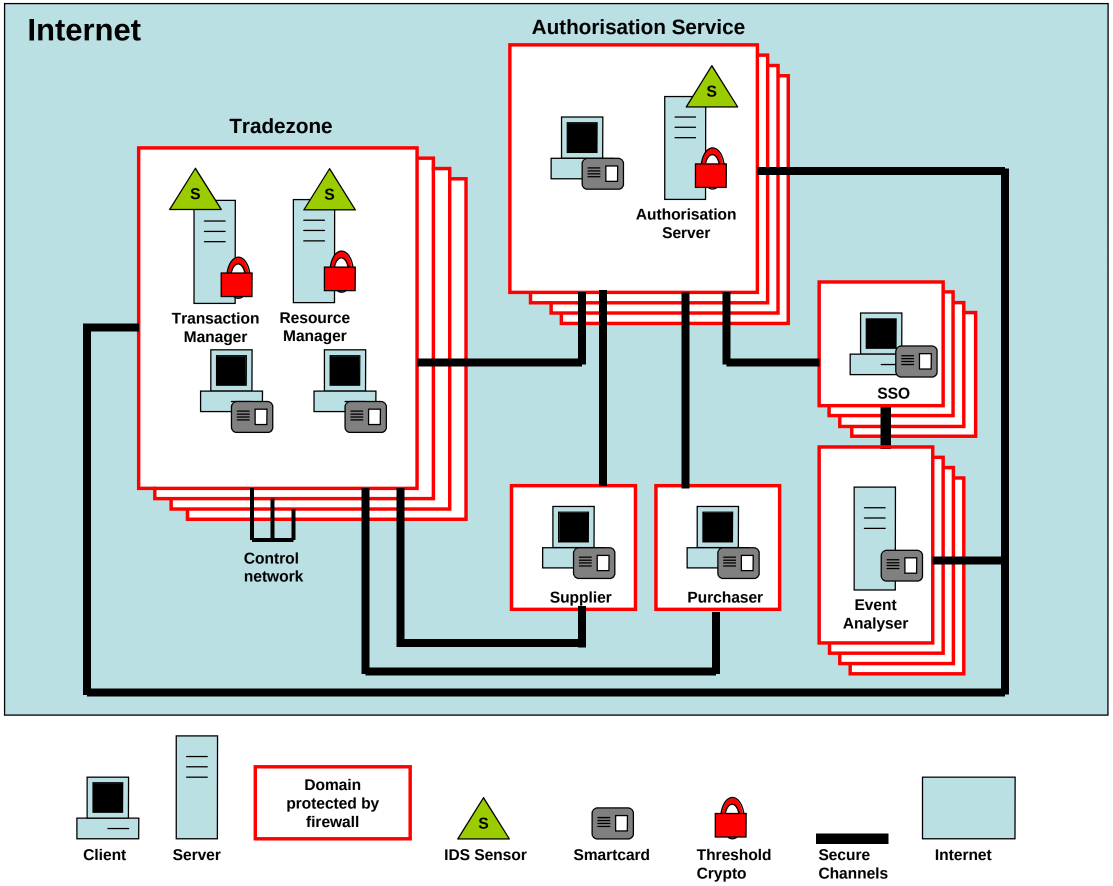
and checked by authorisation checker)

IDS operates in the background

Security failures

- A security failure will occur if one of the security goals of the Tradezone application is violated. For example:
  - A supplier receives payment for which there is no corresponding delivery of goods.
  - Goods are delivered but the supplier does not receive payment.
- Such failures could be due to:
  - flaws in the formulation or implementation of the security rules, allowing unauthorised users to issue order and receipt messages
  - failure of an authentication mechanism, allowing fake messages to be introduced into the system.
  - faults in the architectural assumptions (trusted channels not trustworthy, authorisation mechanisms bypassable)

Tradezone in a MAFTIA setting



Malice - the motivated hacker

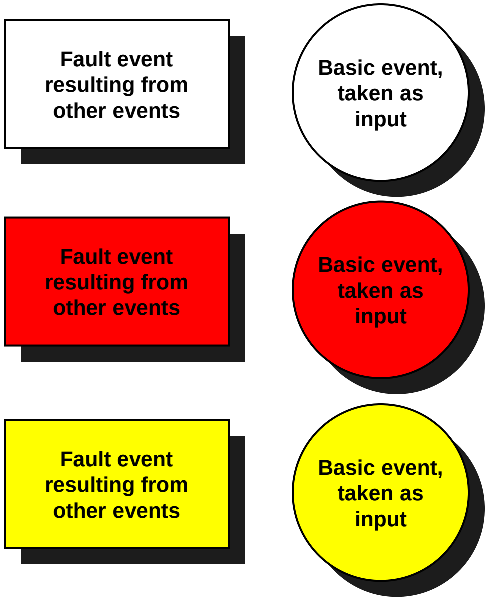
- Malice is a corrupt supplier, and thus an insider with respect to the Tradezone application domain
- Malice's goal is to get money transferred to her account even though she has not performed any service
- We assume that Malice has performed both passive and active reconnaissance of the Tradezone system and has perfect knowledge of its architecture
- However, Malice cannot interfere in interactions between the Broker and the banking system - for example, insert a payment instruction that appears to come from the Broker
- Malice aims to manipulate the payment sequence of messages in such a way that the Broker sends a payment instruction to the banking system that will result in money appearing in Malice's account
- The following fault trees capture possible ways that Malice might try to do this and shows where MAFTIA's intrusion defences come into play

Fault Tree notation

Fault events

MAFTIA
Fault events

IPSec
Fault events



Fault event resulting from other events

Basic event, taken as input

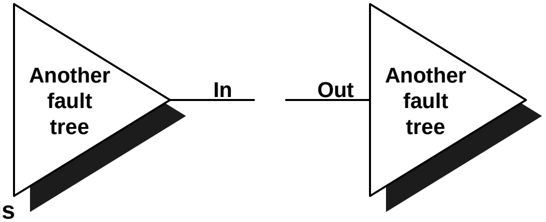
Fault event resulting from other events

Basic event, taken as input

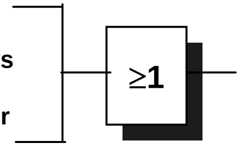
Fault event resulting from other events

Basic event, taken as input

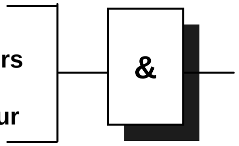
Triangle symbol used to link trees. The ‘in’ input indicates input from another tree, the ‘out’ symbol appears in place of the ‘top event’ and indicates that this point forms the input to another tree



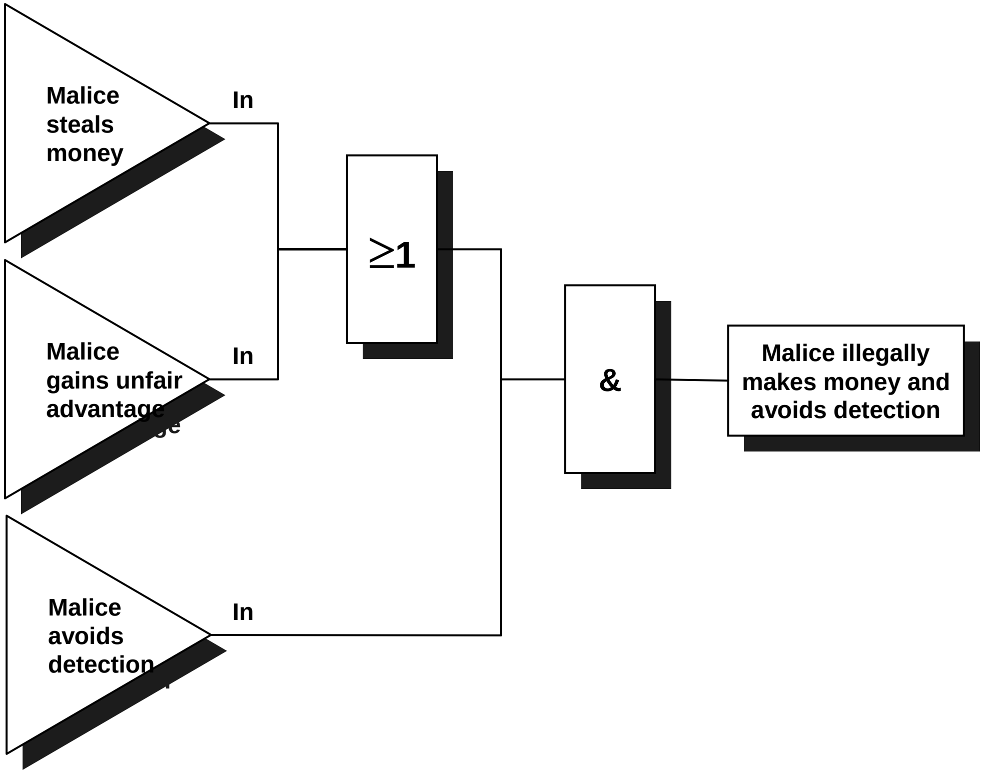
The output event occurs if any of the inputs occur



The output event occurs if all of the inputs occur

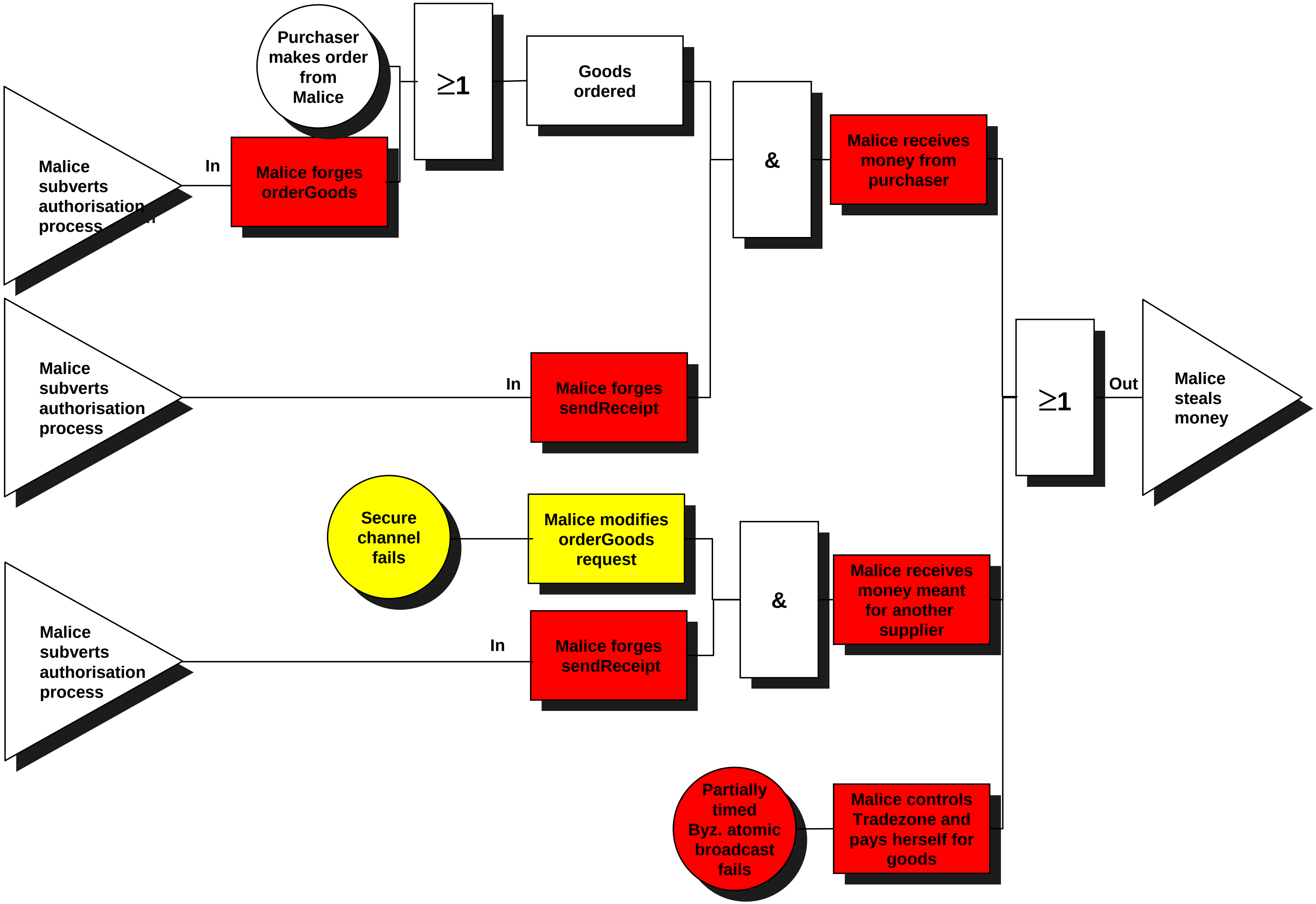


Top-level fault tree


Malice steals money
In
Malice gains unfair advantage
In
Malice avoids detection
In
≥1
&
Malice illegally makes money and avoids detection

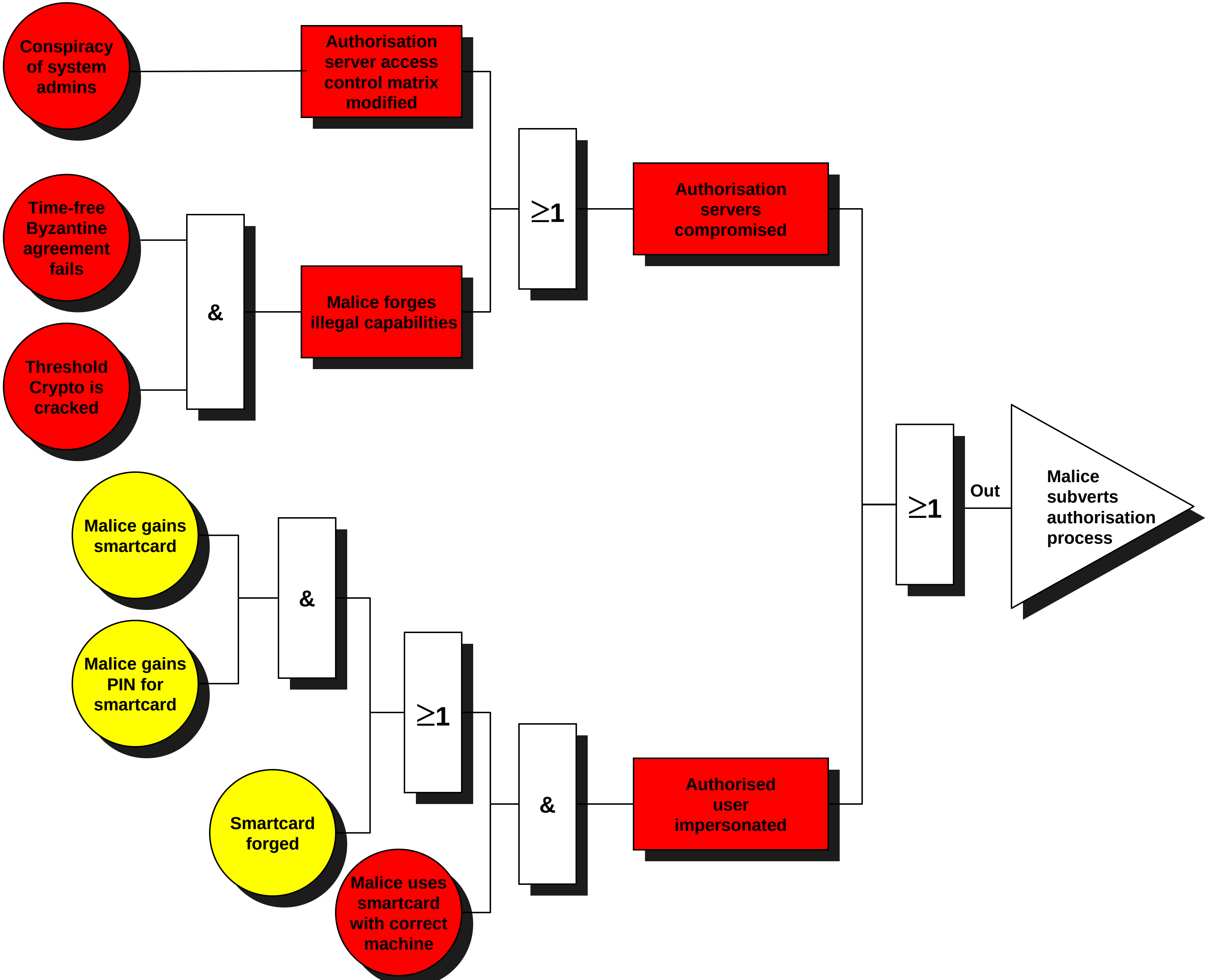
Malice steals money

- We consider three ways in which Malice could steal money from purchasers:
  - She receives money from a purchaser to which she is not entitled
  - She receives money meant for another supplier
  - She subverts the Tradezone application entirely
- Note that we exclude the possibility that Malice can interact with the bank directly, because we assume that the channels between Tradezone and the banks are physically secure and not accessible to outsiders.
- Malice is an insider with respect to the Tradezone application domain, but an outsider with respect to the Tradezone administrative domain


Malice subverts authorisation process
In
Malice forges orderGoods
Purchaser makes order from Malice
≥1
Goods ordered
&
Malice receives money from purchaser
Malice subverts authorisation process
In
Malice forges sendReceipt
≥1
Out
Malice steals money
Secure channel fails
Malice modifies orderGoods request
&
Malice receives money meant for another supplier
Malice subverts authorisation process
In
Malice forges sendReceipt
Partially timed Byz. atomic broadcast fails
Malice controls Tradezone and pays herself for goods

Malice subverts authorisation

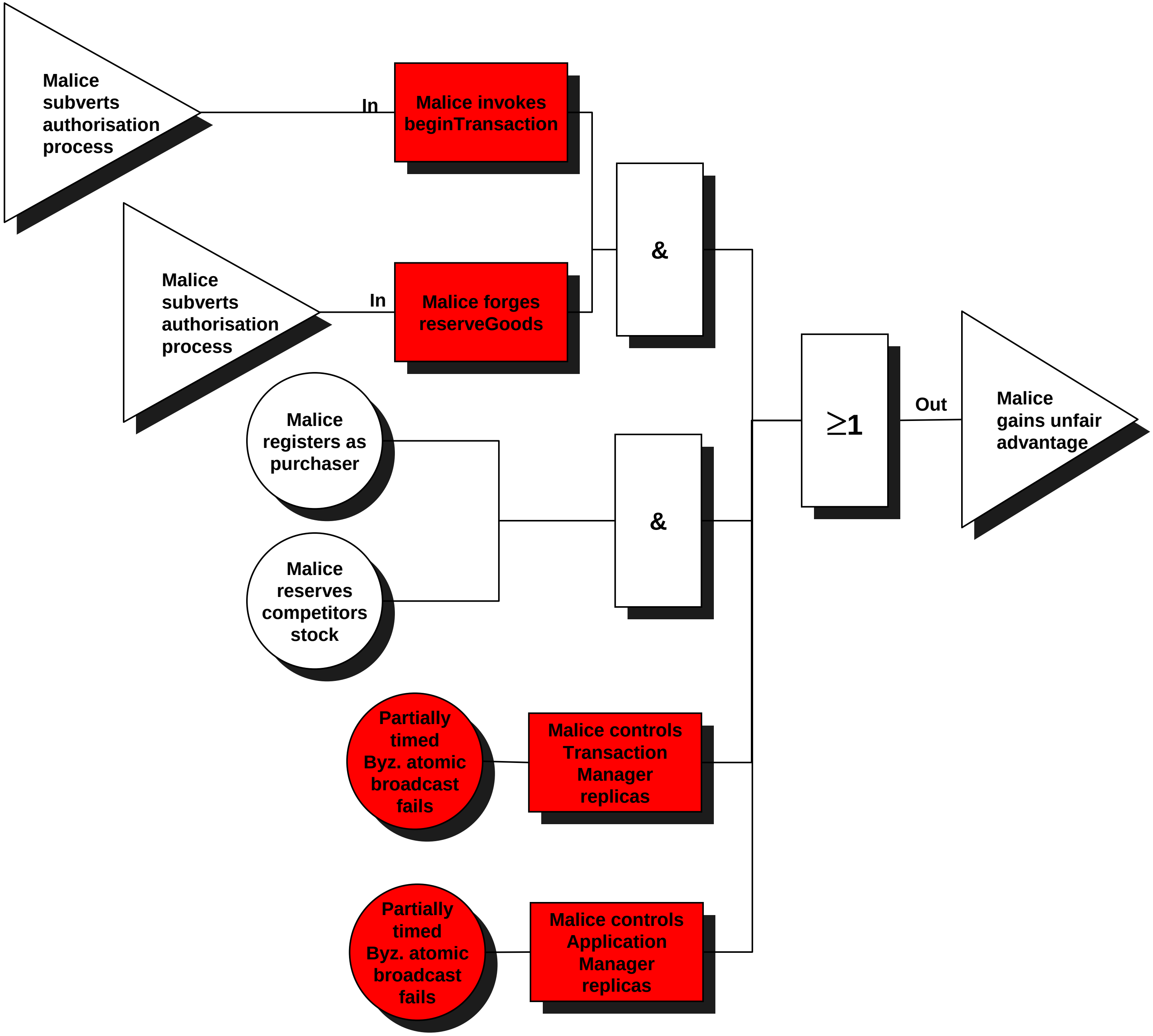
- The cases considered are as follows.
  - Authorisation service compromised
    - Malice modifies the authorisation service’s access control matrix.
    - Malice forges illegal capabilities.
  - Authorised user impersonated
    - Malice steals the purchaser’s smartcard and obtains its PIN.
    - Malice forges a user’s smartcard
- Note that we do not consider the possibility that the authorisation mechanism on the local host can be bypassed
- The authorisation scheme is designed to ensure that even if a particular host is corrupted, it cannot persuade a non-corrupted host to execute unauthorised operations on its behalf.



Conspiracy of system admins
Authorisation server access control matrix modified
Time-free Byzantine agreement fails
Threshold Crypto is cracked
&
Malice forges illegal capabilities
≥1
Authorisation servers compromised
Malice gains smartcard
Malice gains PIN for smartcard
&
Smartcard forged
≥1
Malice uses smartcard with correct machine
&
Authorised user impersonated
≥1
Out
Malice subverts authorisation process

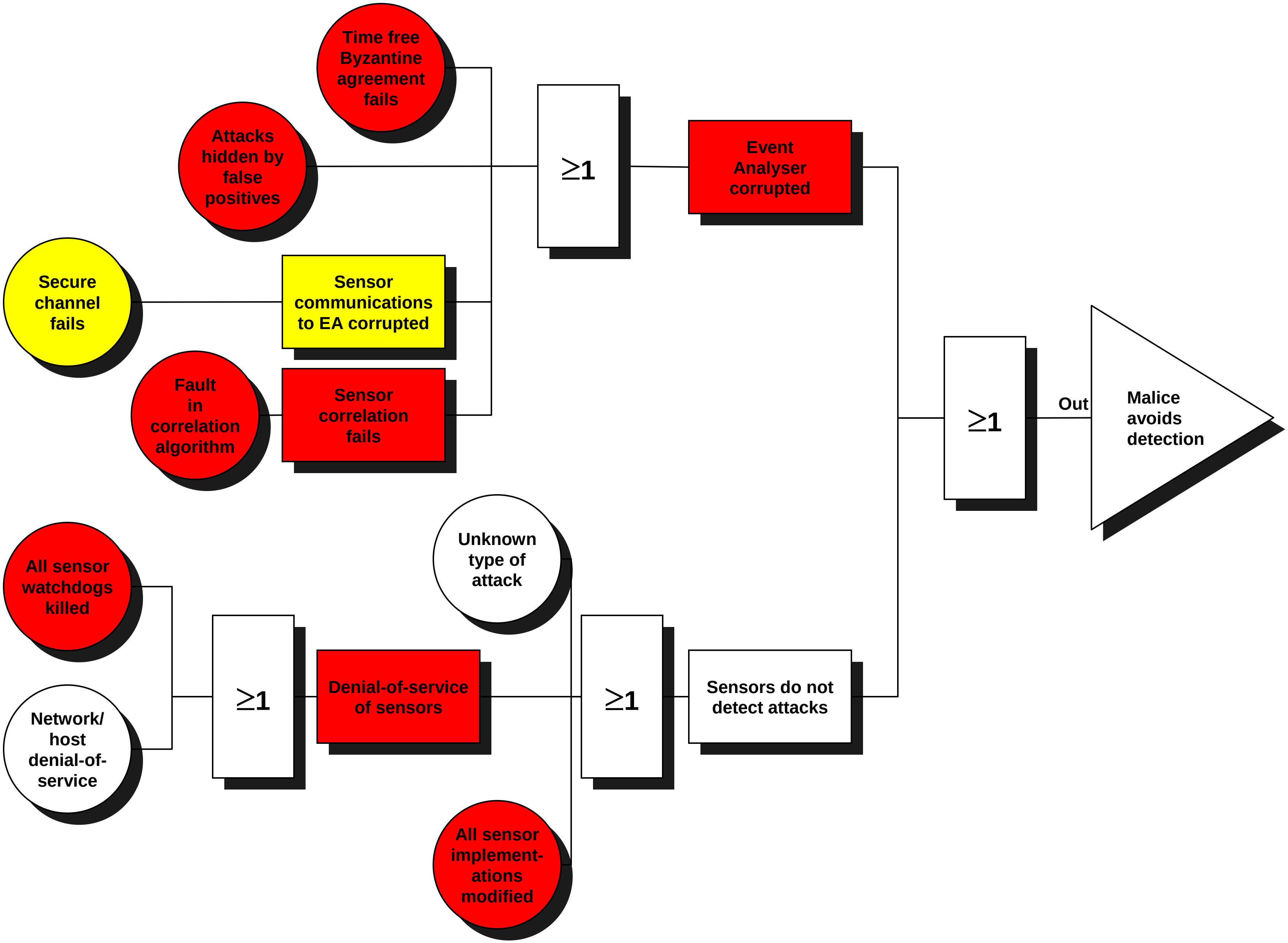
Malice gains an unfair advantage

- Malice can also steal money from her competitors indirectly…
- She could attempt to manipulate the Tradezone marketplace so as to ensure that her competitors cannot offer the same goods at a cheaper price:
  - Malice could act as a purchaser (legitimately or illegitimately) and reserve all available stock from her competitors
  - Malice could attack the integrity of the catalogues and alter the prices
  - Malice could subvert the Tradezone application and deny access to her competitors' products in this way


Malice subverts authorisation process
In
Malice invokes beginTransaction
Malice subverts authorisation process
In
Malice forges reserveGoods
&
Malice registers as purchaser
Malice reserves competitors stock
&
≥1
Out
Malice gains unfair advantage
Partially timed Byz. atomic broadcast fails
Malice controls Transaction Manager replicas
Partially timed Byz. atomic broadcast fails
Malice controls Application Manager replicas

Malice evades detection

- Malice would aim to avoid detection by the Intrusion Detection System (IDS) at least while she was carrying out her activities:
  - She could corrupt the event analysers, and thus prevent the IDS from correctly detecting her attack
  - She could ensure that events associated with her activities were not collected in the first place
- Note that we only consider technical attacks on the IDS, and do not consider the possibility that Malice could conceal her activities by masquerading as a legitimate user


Time free Byzantine agreement fails
Attacks hidden by false positives
Secure channel fails
Sensor communications to EA corrupted
Fault in correlation algorithm
Sensor correlation fails
≥1
Event Analyser corrupted
≥1
Out
Malice avoids detection
All sensor watchdogs killed
Network/ host denial-of-service
≥1
Denial-of-service of sensors
Unknown type of attack
≥1
Sensors do not detect attacks
All sensor implement-ations modified

Discussion

- An intrusion-tolerant system must be able to continue to deliver a secure service, despite the presence of intrusions
- Hence, a “defence in depth” strategy is needed so as to avoid depending on any particular component of the system that could become a single point of failure
- It is important to be clear about which components of the architecture are trusted, and to what extent
- MAFTIA uses a variety of techniques for distributing trust and limiting trust

Preventing errors from leading to security failures

- An error that could result in a security failure is not necessarily the result of a malicious fault
- Hence, it is important to deal with accidental faults as well as malicious faults
- Using state based error recovery approaches to achieve intrusion tolerance is problematical for two reasons:
  - Need to be able to detect errors correctly with high probability
  - Need to be able to deal with latency of error detection and long-lived dormant faults
- Thus, MAFTIA’s approach to error handling is to use error compensation techniques based on active replication and fault masking.
- Hence, Byzantine agreement protocols are at the heart of the intrusion tolerance mechanisms provided by MAFTIA.

The role of diversity

- Although Byzantine agreement protocols are designed to deal with arbitrary faults in the value and time domain, they still assume failure independence.
- In the presence of malicious faults and deliberate attacks on systems, this is not a reasonable assumption,
- MAFTIA must therefore take steps to address this problem by designing protocols that can tolerate more realistic failure assumptions, and by ensuring that compromising one replica does not make it any easier to compromise another.
- The standard way of achieving failure independence is to make some assumption about diversity, and thus, various forms of diversity are an important part of any intrusion tolerance strategy.

Distributing trust (fault masking)

- MAFTIA has explored two different approaches to implementing fault masking using active replication.
- The TTCB approach uses fault prevention techniques to build a trustworthy TTCB, which can then be used to support the execution of fault-tolerant protocols.
- The asynchronous approach uses cryptographic algorithms to implement an efficient probabilistic Byzantine agreement protocol, and does not make any assumptions about the trustworthiness of individual components or hosts.
- In one case, the TTCB is assumed to be tamper-proof by design
- In the other case, a generalised adversary structure is used to model a more realistic set of failure independence assumptions

Limiting trust (error confinement)

- “Defence in depth” involves accepting that components of the system can be compromised,
- Thus, it is important to limit the extent to which components are trusted by other components and prevent intrusion propagation
- Hence, error confinement strategies are an important part of achieving intrusion tolerance
- There are many examples of such strategies within MAFTIA:
  - Use of threshold signature schemes
  - The distributed authorisation scheme assumes that hosts are mutually suspicious of each other
  - Using diverse sensors and event analysers for intrusion detection

Summary

- We have constructed a “use case” for the MAFTIA architecture based on a small but realistic e-commerce application
- A fault tree analysis has been used to highlight some of the obstacles that an attacker needs to overcome in order to intrude upon the system
- The analysis illustrates a number of key design principles for building intrusion tolerant systems:
  - the use of a “defence in depth” strategy,
  - recursive use of fault prevention and fault tolerance
  - the notion of trusting components to the extent of their trustworthiness but no more
  - measures to increase trust by distributing it across a set of replicas
  - measures to limit trust so as to prevent intrusions from propagating throughout a system.

Future research

- MAFTIA has concentrated on achieving intrusion tolerance using error compensation mechanisms based on masking.
- However, the success of these measures depends on failure independence assumptions, which are typically based upon claims about diversity.
- Thus, three important areas for future research are:
  - developing techniques for ensuring and measuring diversity in the presence of arbitrary malicious faults,
  - improving the quality of error detection mechanisms so as to make state-based error recovery techniques feasible as a means of intrusion tolerance, and finally,
  - finding solutions to the problems of long latency of error detection and malicious dormant faults