Security Beyond the Libraries
(or the story of Alice and Bob)

Eoin Woods
Artechra
www.eoinwoods.info

v2 20141015

Introduction

Introduction

- Security is a difficult thing to achieve
  - becoming more important all the time
- Development teams often approach it via technologies
  - “SSL” “Spring Security” “SSO” “Roles” “OAuth”
  - “FindBugs” “FxCop” “Fortify” “Tripwire”
- This is completely the wrong way around
  - you need to understand your risks before finding solutions
  - technology is only part of the solution
- In this talk we discuss how to base security on risks and the principles for designing (more) secure systems

Caveats

- This talk is introductory in nature
  - some things aren't talked about at all
  - some things are just introduced
- This talk is for system designers not security engineers
  - there are many subtleties that are skipped over
  - some things are simplified to their essentials
- I'm not a security engineer!
  - I have worked in a secure systems environment
  - I do have reasonably good security knowledge
  - But I'm not a specialist
- You still need a security engineer!

Introducing Security

The Need for Security

- We need systems that are dependable in spite of
  - Malice
  - Error and
  - Mischance
- People are sometimes bad, stupid or just unlucky
- System security attempts to mitigate these situations

The Need for Security

- Systems threats are similar to real-world threats:
  - Theft
  - Fraud
  - Destruction
  - Disruption

- Anything of value may attract unwelcome attention

"*I rob banks because that's where the money is*"
– Willie Sutton

The Need for Security

- Why do we care about these threats?
  - A threat is the risk of a loss of some sort
- Kinds of loss include:
  - Time
  - Money
  - Privacy
  - Reputation
  - Advantage

What is Security?

- Security is the business of managing these risks
  - Security is a type of insurance
- Balances cost and effort against risk of loss
- Some basic terminology
  - resources - things of value that (may) need protection
  - actors - good and bad people interacting with the system
  - policies - the rules to control access to the resources
  - threats - the reason that the rules may be broken
- A common cast of characters
  - Alice, Bob, Eve and Mallory

What is Security?

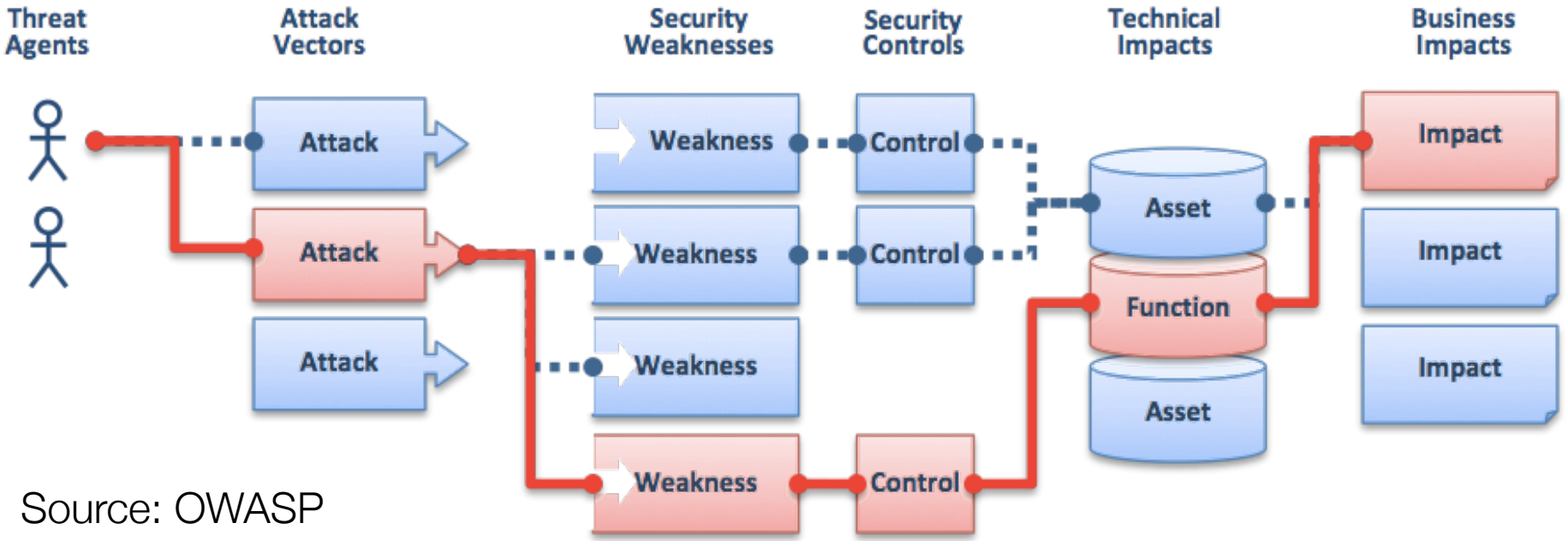
- Security is multi-dimensional
  - People
    - Users, administrators, security experts (and of course, attackers)
  - Process
    - Design, operation, control, monitoring, …
  - Technology
    - What to apply, how to use it, how to integrate it

Security is not a product -- it's a process
Bruce Schneier

You're only as strong as your weakest link!

A Word on Risks, Threats and Attacks

- Commonly confused terminology
  - Vulnerability = a weakness in a security mechanism
  - Threat = Vulnerability + Attacker + Motivation
  - Attack = when the attacker puts a plan into action
  - Risk = threat x likelihood x impact



Source: OWASP

Key Security Requirements

- Confidentiality (or Privacy)
  - Prevent unauthorised access to information
- Integrity
  - Prevent tampering or destruction
- Availability
  - Prevent disruption to users of systems
- Accountability (or “non-repudiation”)
  - Know who does what, when

Confidentiality, Integrity & Availability = “CIA Triad”
Authentication, Authorisation & Accounting = “AAA”

Overview of Security Mechanisms and Attacks

Security Mechanisms and Attacks

- This talk is for system designers, not security specialists
- Understanding mechanisms and attacks is hard
  - people who like this sort of thing are security engineers!
- This section overviews key mechanisms and some of the more important types of attack
  - this isn't a detailed explanation of any of this
  - make sure you consult a security expert when needed
  - *this* is useful to understand the basics and the language

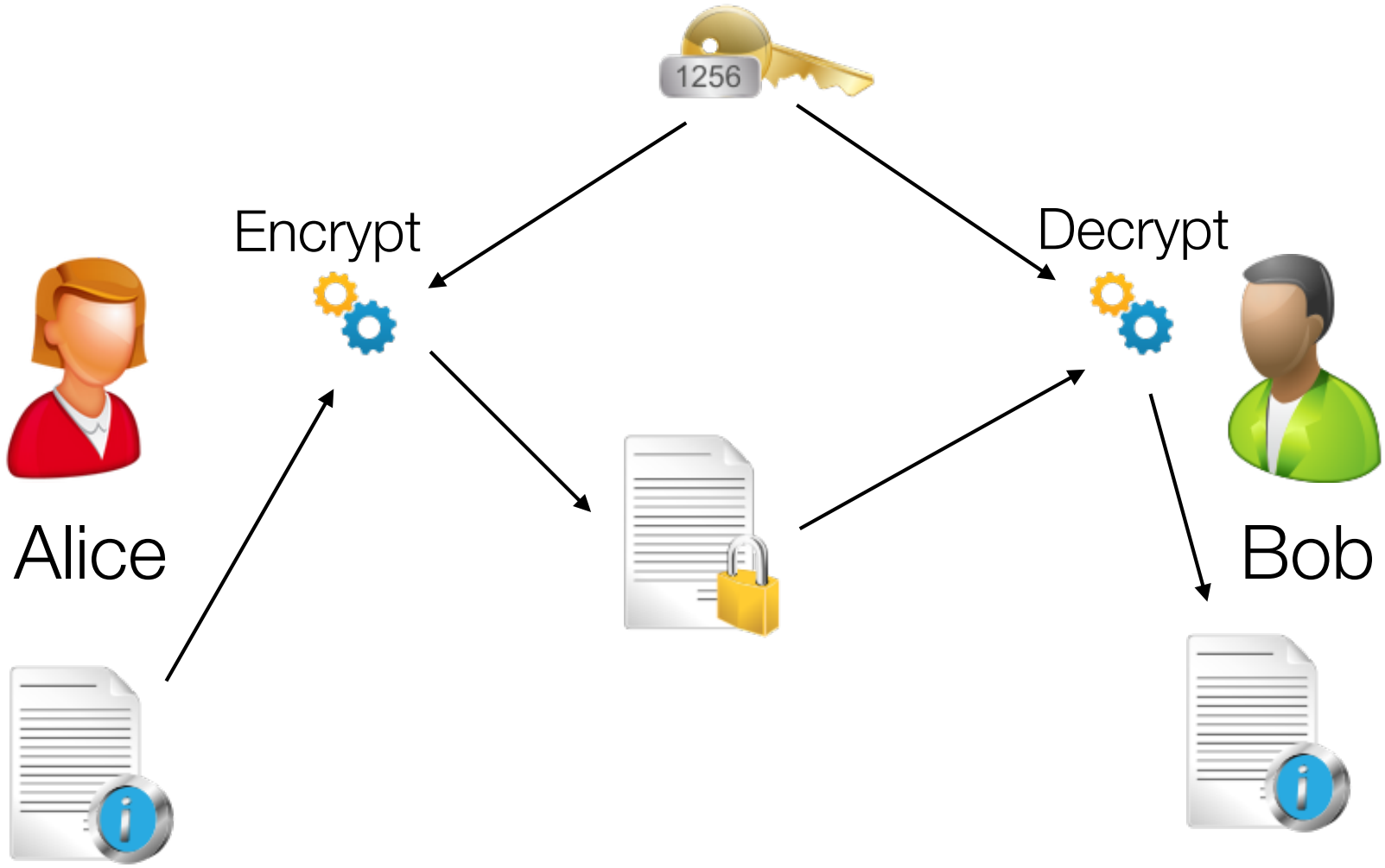
Partial Summary of Security Mechanisms

Authentication ("Who are you?)	Usernames & passwords, 2FA, biometrics, certificates, …
Authorisation ("What can you do?")	Roles, access control lists, OAuth, permissions, …
Confidentiality ("Keeping stuff secret")	Encryption, access control lists, …
Integrity ("Stop tampering")	Cryptographic hashing, checksums, …
Non-Repudiation ("You did that")	Cryptographic signing, audit trails, …
Auditing ("What happened, when?")	Secure record of who did what, when

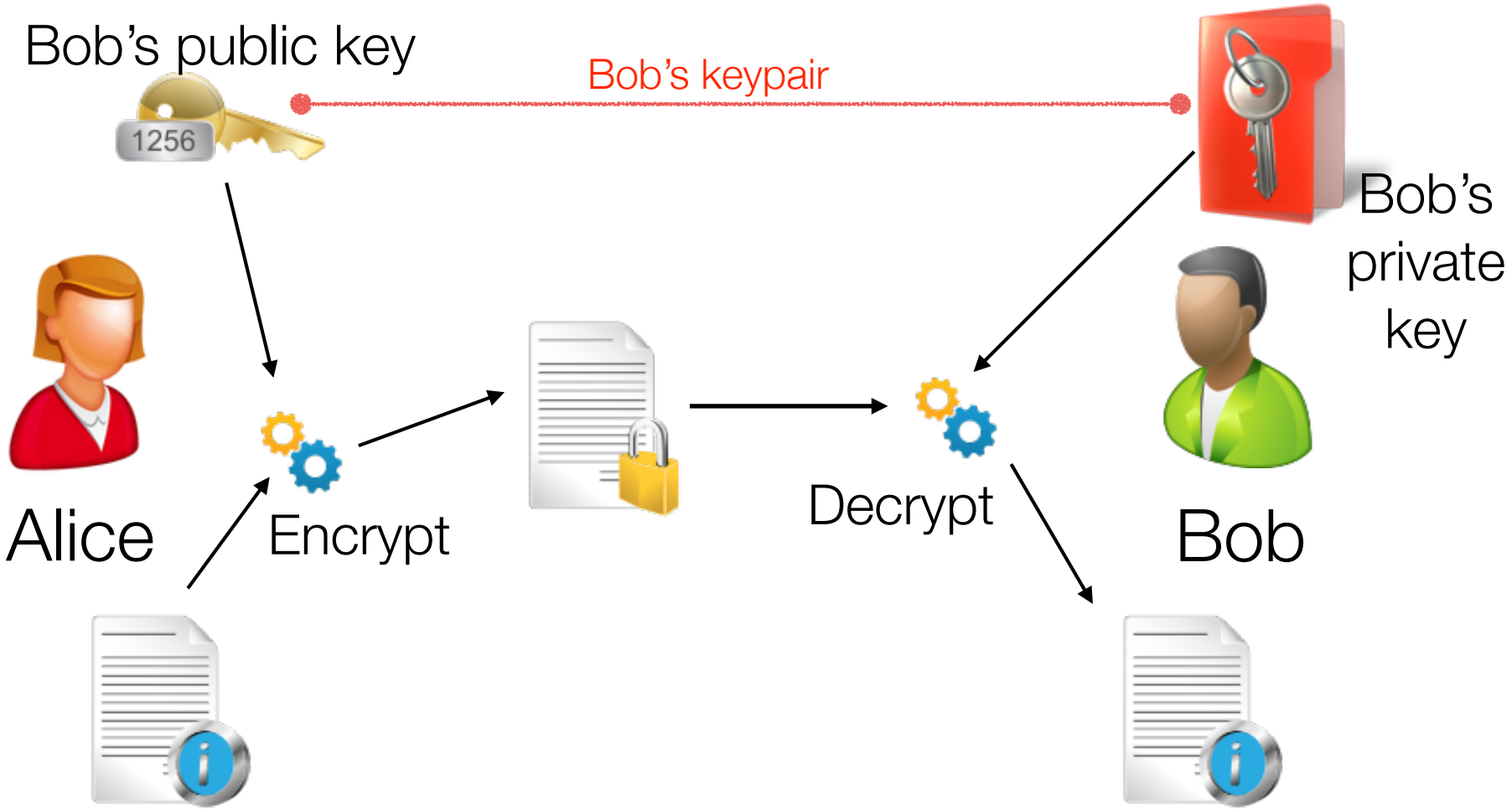
Cryptography and its Uses

- Cryptography: the process of keeping information private (typically) using a cypher and a key
- Two broad subfields:
  - Symmetric Key - the same key to encrypt and decrypt
  - Public Key - a pair of keys one to encrypt one to decrypt
  - Often used together for real applications
- Applications of cryptography: privacy (encryption), integrity (signing) and non-repudiation (signing)
- Difficulties with cryptography include designing good ciphers, key management and secure design

Basic Symmetric Key Cryptography



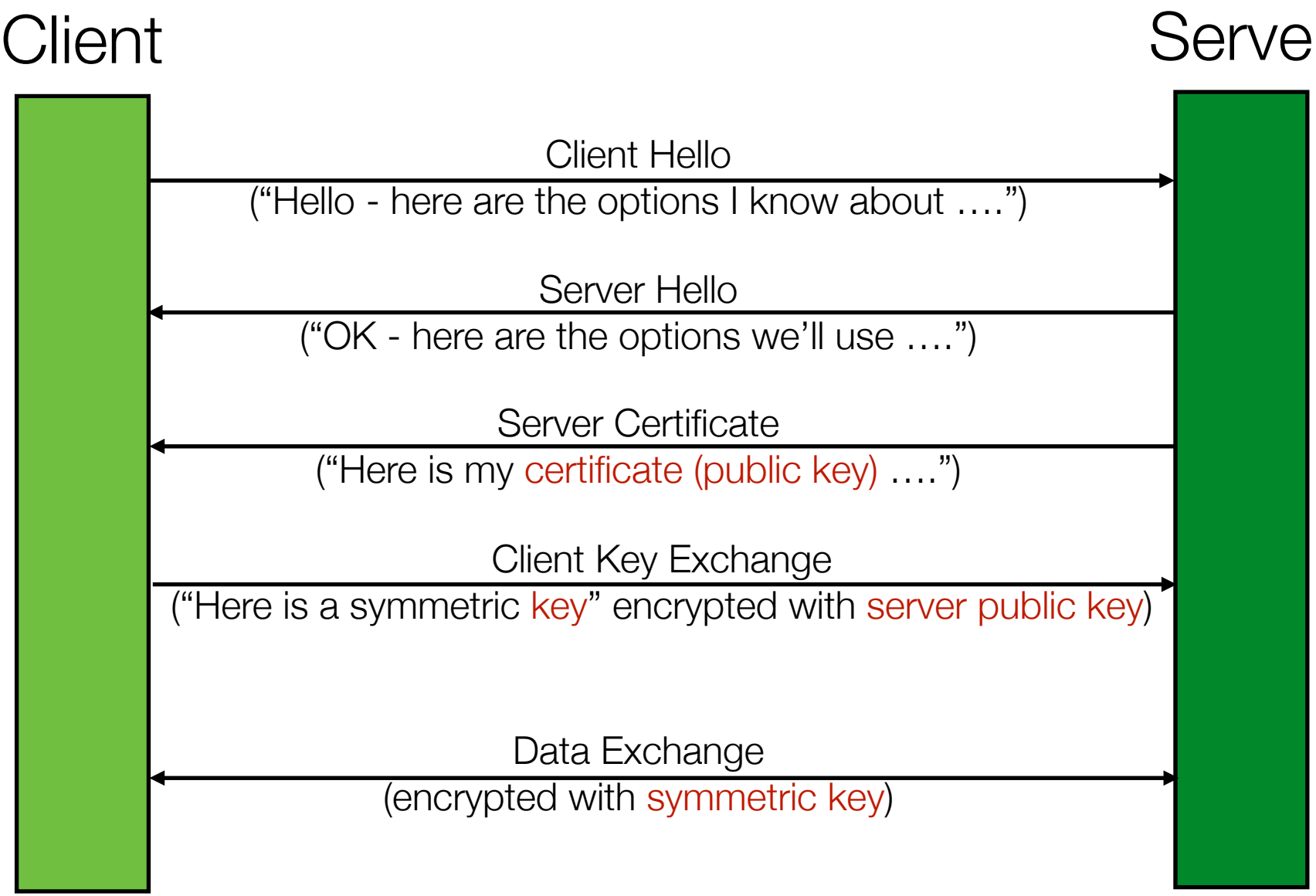
Basic Public Key Cryptography



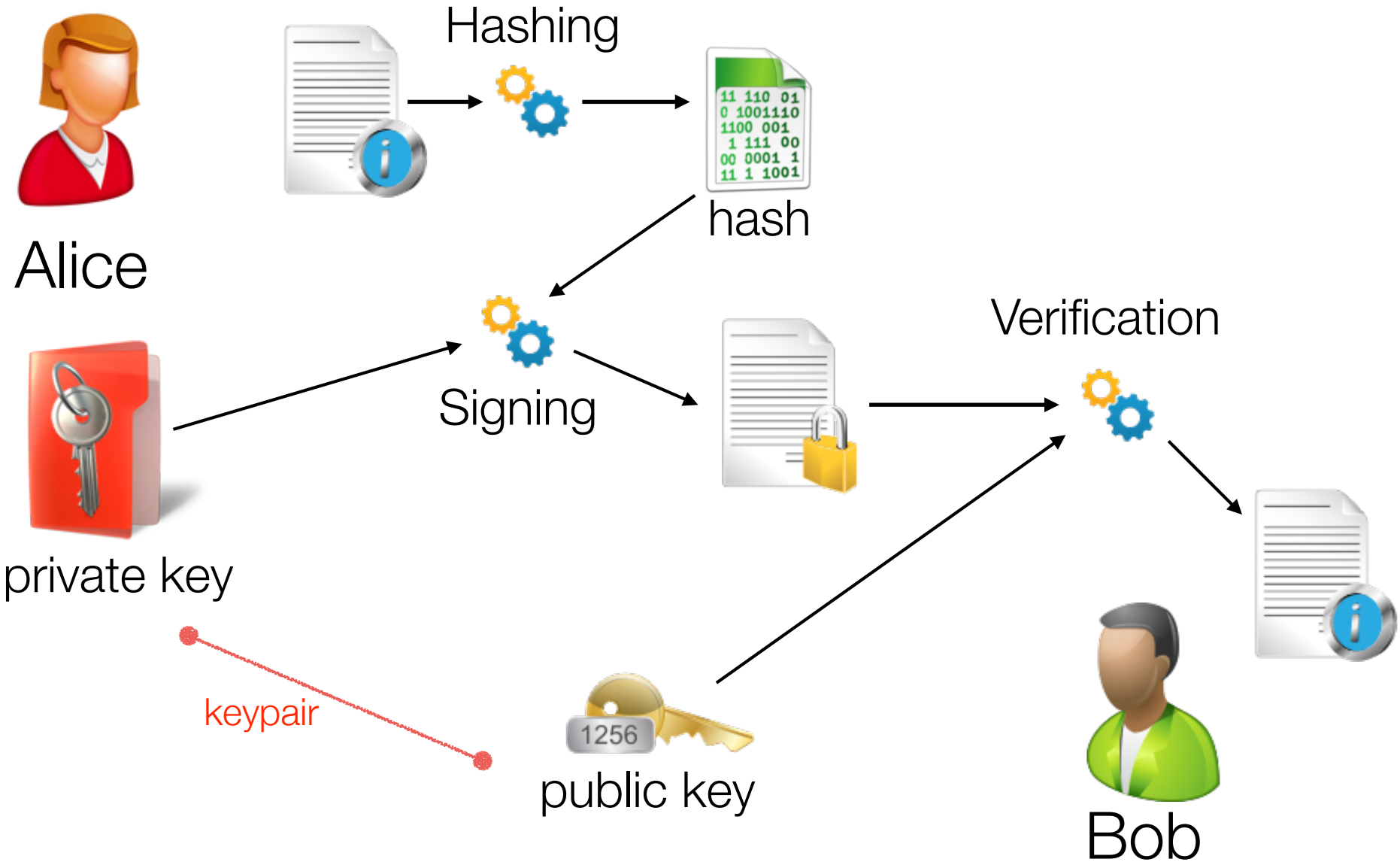
Comparing Symmetric and Public Key

Symmetric Key	Public Key
Single shared key	A keypair needs to be generated with public and private key
Single key needs to be shared securely	Private key is kept secret, public key can be freely shared
Efficient computationally	Really slow (~100s times slower)
Classic attack is stealing keys	Classic attack is forging public keys, so faking identity
Big challenge is passing keys around	Big challenge is chains of trust so you know keys (certs) are valid

Example of Crypto in Practice: Key Exchange



Hashing and Signing



Summary of Common Attacks

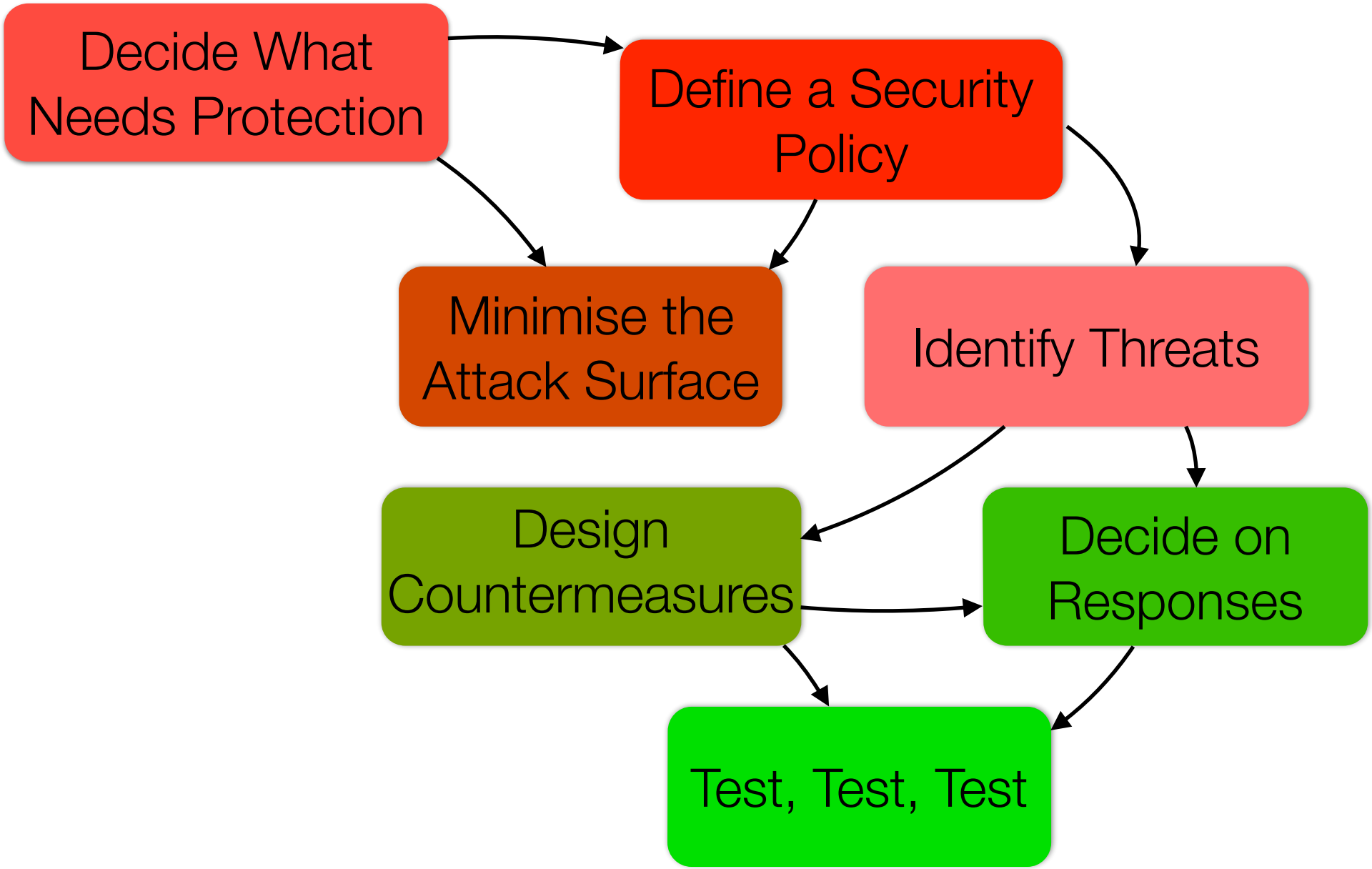
Social Engineering (Attack the people)	Bluffing, bribery, coercion, …
Network Attacks (Protocol exploits)	SYN floods, DDoS, DNS attacks, …
Application Attacks (Flaws and bugs)	XSS, SQL injection, corrupted input (fuzz), DLL attacks, CGI/SSI attacks …
Programming Attacks (Abuse of a language)	Stack overflows, buffer overflows, known bugs in libraries, …
Crypto Attacks (Algorithms or design)	Known plain/cipher text, randomness, cert forgery, man in the middle, traffic analysis…

Attacks on Secure Systems

- A lot of work goes into analysing cryptoanalytic attacks so social engineering is usually more efficient
  - though you do need to know your crypto is sound too
- When analysing attacks consider the goals and capabilities of an attacker
  - script kiddies don't crack crypto
  - governments crack crypto on demand (NSA, Chinese, …)
- As *system* designers we need to understand the threat an attack implies, not every detail of how it works
  - your security engineers will understand the details
- Always remember new attacks appear constantly!

Securing Systems

Securing a System



Securing a System

- Decide what needs to be protected
  - Identify resources valuable to us and potential for loss
- Specify a security policy to protect our resources
- Understand the threats the policy faces
- Implement mechanisms to counteract the threats
- Use an assurance process to test our security

Identifying Valuable Resources

- What is valuable is often self-evident
  - but think like an external attacker
  - things of low value to you might be useful to them
  - e.g. configuration files … reveal network topologies
  - e.g. client information … damaging if lost
- Think operations as well as data
  - e.g viewing a payment might be fine
    … creating and releasing a payment is another question!
- May require fine-grained consideration
  - e.g. classic example is HR data where phone numbers are low sensitivity, home address is much more sensitive

Defining Security Policy

- Security policy is effectively the security specification
  - controls and guarantees needed in the system
  - types of principal who will use the system
  - the resource types within the system
  - the actions to be performed on each resource type
  - the rights of each type of principal in terms of actions on resources

Security Policy

	Clients	Orders	Refunds <= £100	Refunds > £100
Onshore Service Agents	Create, View, Modify (Un)Suspend	All	Create, View, Authorise	View
Offshore Service Agents	View, (Un)Suspend	View, Cancel	View	View
Supervisors	All	All	All	Create, View, Cancel
Finance	View	View	View, Authorise	All

Minimise the Attack Surface

- The attack surface is the set of potentially vulnerable ways into of the system (“attack vectors”)
  - smaller attack surface means less to attack
  - smaller attack surface means less to secure
- OWASP define the attack surface as:
  - all channels into and out of the system
  - the code securing those channels
  - data of value within the application (security and domain)
  - the code securing this data
- Reducing interfaces, protocols, services, …
  - which conflicts with other properties you may want!

Understanding Threats

- Threat is the possibility of a breach in security policy
  - System/process/people may (will) have vulnerabilities
  - Attackers have motivation and goals
  - Threat is the danger of the attacker exploiting a vulnerability
- Identifying threats is a key part of security design
  - threats are where you focus your security effort
  - threat modelling is the key activity

Understanding Risks and Threats

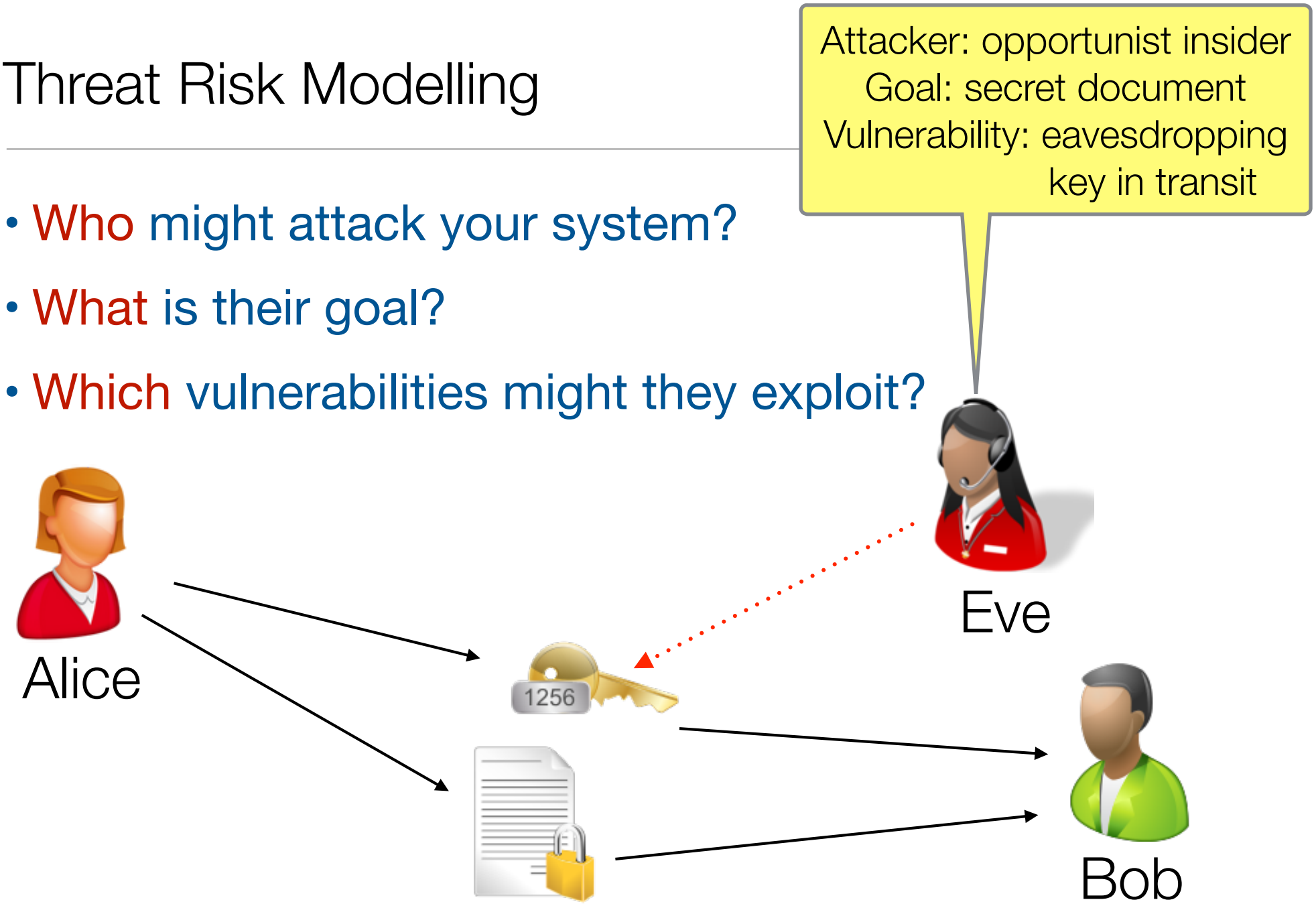
- Recap: Threats and Risks
  - risk is a vulnerability that *could* be exploited
  - threat is an attacker with a motive leading to an attack
- Model risks as a Threat Risk Model
  - or just "*threat model*" in some cases
- Model threats as an Attack Tree
- Use Abuse Cases to think about how an attacker will attack the system and the likely outcome

Threat Risk Modelling

- Threat modelling - *a procedure for optimizing security by identifying* objectives *and* vulnerabilities, *and then defining* countermeasures *to prevent, or mitigate the effects of, threats to the system* — OWASP
- Goal is to identify the real risks to focus security effort
- Useful Microsoft checklist:
  - Spoofing
  - Tampering
  - Repudiation
  - Information disclosure
  - Denial of service
  - Elevation of privilege

Threat Risk Modelling

- Who might attack your system?
- What is their goal?
- Which vulnerabilities might they exploit?

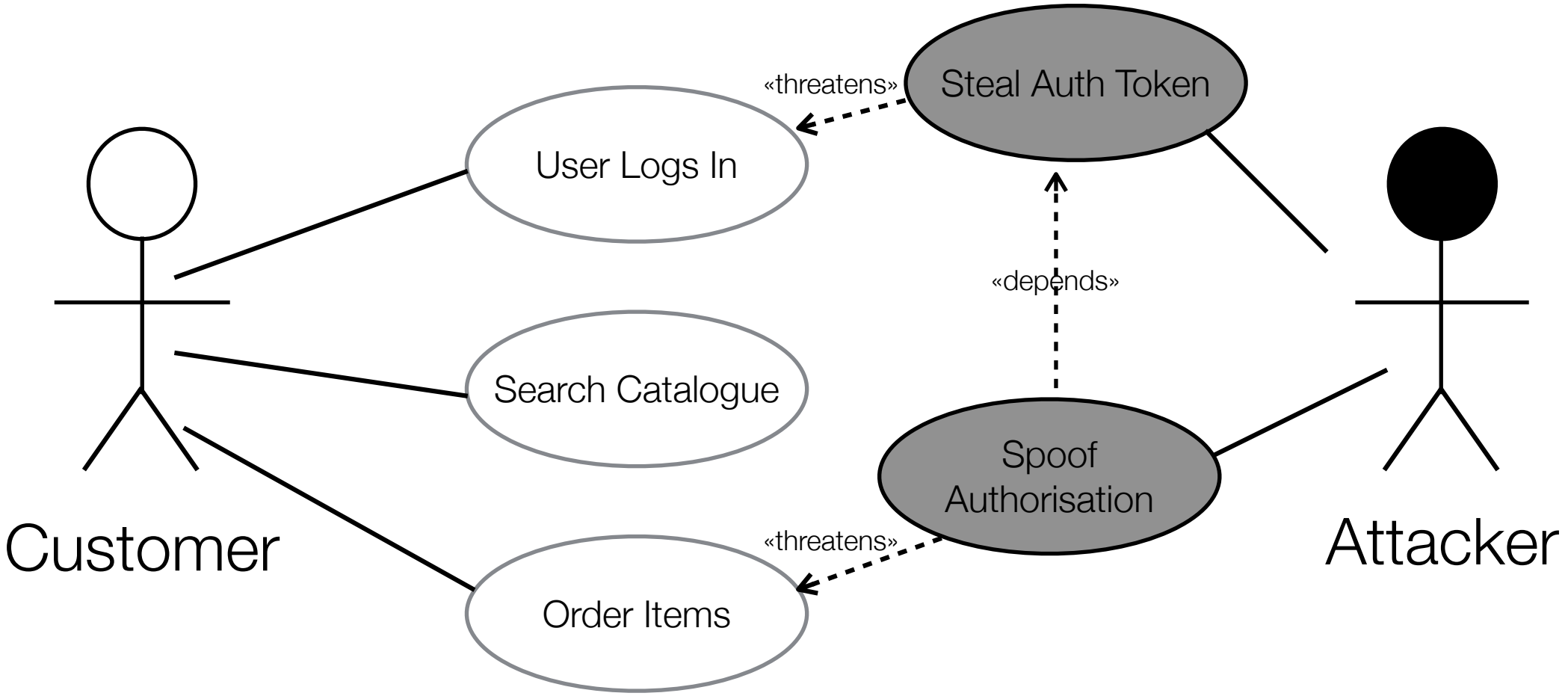


Attack Tree Example

Attacker: Professional hacker
Goal: Obtain customer credit card details
Attack: Extract details from the system database.

1. Access the database directly.
   - 1.1. Crack/guess database passwords.
   - 1.2. Crack/guess operating system passwords that allow database security to be bypassed
   - 1.3. Exploit a known vulnerability in the database software
2. Access the details via a member of the database administration staff.
   - 2.1. Bribe a database administrator (DBA).
   - 2.2. Conduct social engineering by phone/e-mail to trick the DBA into revealing details.
3. …

Security Abuse Cases



Abuse Case Example

Abuse Case:	Spoofing Authorisation via Valid Authentication
Threat:	The misuser steals an authorisation token and attempts to use it via a valid (other) authenticated identity
Preconditions:	1) The misuser has a valid means of user authentication (e.g. username/password). 2) The misuser has a stolen user authorisation token.
Actions:	1. The system shall request the user's identity and authentication. 2. The misuser authenticates himself correctly. 3. The system shall identify and authenticate the user. 4. The misuser attempts to authorise using the stolen token. 5. The system rejects the authorisation attempt, audits the event, terminates the session and locks the user account.
Postconditions:	1. The system shall have identified and authenticated the misuser 2. The system shall have prevented the misuser from stealing another user's means of authorisation.

Assessing Risks

- The DREAD model is useful and straightforward
- Risk = Damage +
  Reproducibility +
  Exploitability +
  Affected Users +
  Discoverability
- Simple numerical scales for each dimension
- Example: damage for individual users, reproducible with a browser, malware required for exploit, many but not all users affected and can be discovered easily … 7/10
  - see https://www.owasp.org/index.php/Threat_Risk_Modeling#DREAD

Designing Mitigations

- Once you have prioritised and understood risks you can design mitigations for them
- Some are pretty obvious and straightforward
  - e.g. use of role based access control to protect resources
- Some are more complex but well known
  - e.g. XSS or SQL injection require careful input validation
- Some need custom solutions
  - e.g. attacks based on organisation process or structure
- Mitigations include people, process and technology!

Incident Response

- However secure your system is you may have a breach
- Need a plan for what you do when it happens
  - an incident response plan
  - a standing incident response team
- Broader than technical mitigation
  - technical, management, legal & communications
- A plan that allows a clear, logical, risk driven response
  - finding the problem, mitigating, collecting evidence, communicating, learning lessons
- Practice your response

Secure Implementation

- Secure design is worthless if implemented insecurely
  - secure implementation outside the scope of this talk
- Like design secure implementation can be complicated
  - requires knowledge and care
  - relatively specialist field
- Static analysis and expert code review
  - FxCop, FindBugs, CodeAnalysis/C++, Coverity, Fortify, …
  - OWASP code review guidelines
  - Oracle Java security guidelines

Testing and Verification

- Like any software quality security needs to be tested
  - security testing largely outside the scope of this talk
- Wide range of security testing activities:
  - static analysis of code
  - functional testing of security features (e.g. authorisation)
  - penetration testing
  - fuzz testing
  - manual system security review
  - known vulnerability tests
  - threat mitigation tests
- Risk driven approach needed to maximise RoI

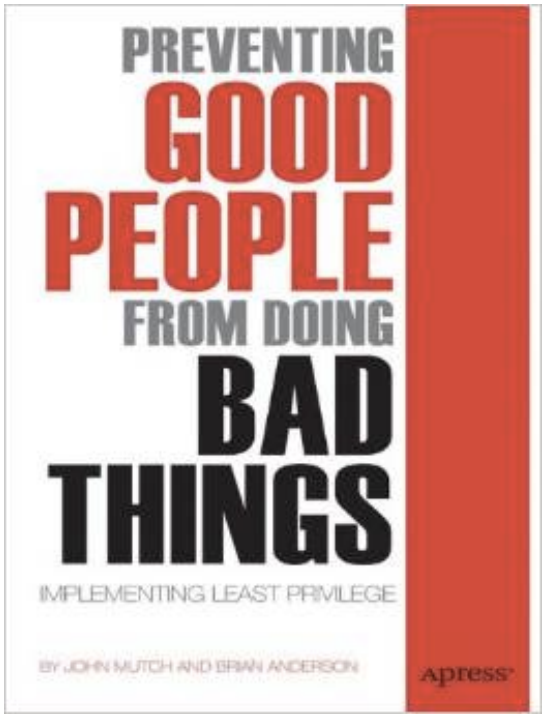
Design Principles for Secure Systems

Design Principles for Secure Systems

- Principles to guide us as system designers:
  - Assign the least privilege possible
  - Separate responsibilities
  - Use the simplest solution possible
  - Audit sensitive events
  - Fail securely & use secure defaults
  - Be careful who you trust
  - Never rely upon obscurity
  - Implement defence in depth
  - Find the weakest link

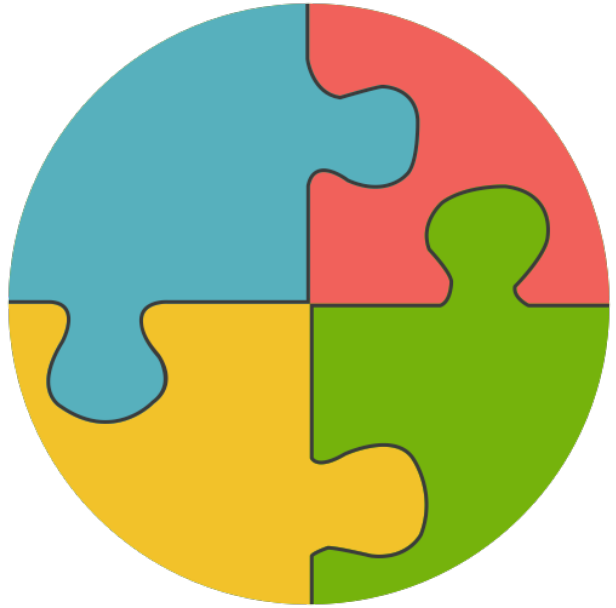
Principle: Least Privilege

- Common problem is broad privileges
- Good principle is to limit to the essentials
  - what do people need to do their jobs
  - anything else is an exception
- Least privilege mitigates malice, error and mischance
- Balance is to still allow people to get their jobs done
- Example: running services as “root” or “Administrator”



Principle: Separate Responsibilities

- Related to least privilege
- Separate and compartmentalise responsibilities and privileges
  - better control and accountability
- Limits the impacts of successful attacks
  - and makes attacks less attractive



- Separate by subsystem, separate by functional role
  - e.g. keep security administration separate from operational functions

Principle: Simplest Solution Possible

- To be secure you need to understand the design to allow analysis
  - A complex design is unlikely to really be understood
- Secure design is difficult, complex secure design is almost impossible!
  - avoid complex failure modes
  - avoid implicit behaviour
  - avoid unnecessary features
- *The price of reliability is the pursuit of the utmost simplicity* - C.A.R. Hoare

Principle: Audit Sensitive Events

- Record all significant security events
- An audit log acts as:
  - a record of activity in the system
  - a deterrent to wrong doing
  - a debug log to reconstruct what happened
  - a monitoring point to spot potential problems

- Remember to build security for the audit log into the policy model and the implementation
  - don’t allow “super users” to alter the audit log

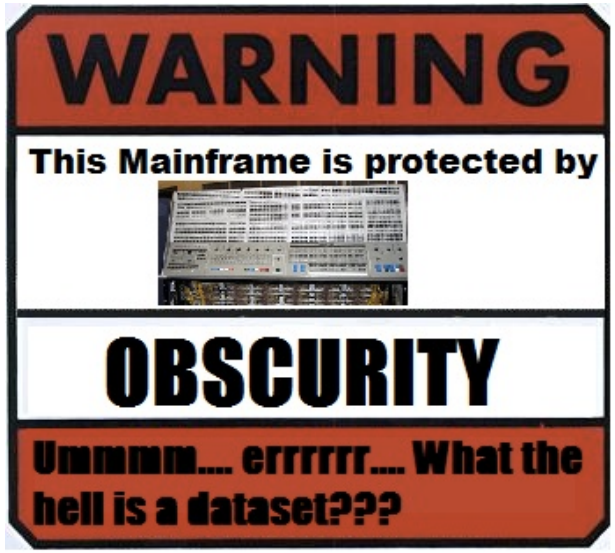
Principle: Use Secure Defaults and Fail Securely

- Remember that default settings usually aren't changed
  - switching off security must be a decision
  - e.g. default network connections over secure transport
  - e.g. highlight insecure defaults if they are used
- Make sure that failure modes end up in a secure state
  - e.g. don't suspend auditing if the log fills up
  - e.g. tear down security context if connection is dropped
- This principle often conflicts with other properties like scalability and availability

Principle: Be Careful Who You Trust

- Assume that unknown entities are untrusted
  - have a clear process to establish trust
  - e.g. don’t assume who is connecting
- Be careful of intermediate entities in transactions or communication paths
  - e.g. man-in-the-middle attacks
  - ensure end-to-end security
- Don’t blindly trust 3rd party components
  - e.g. check signatures on code

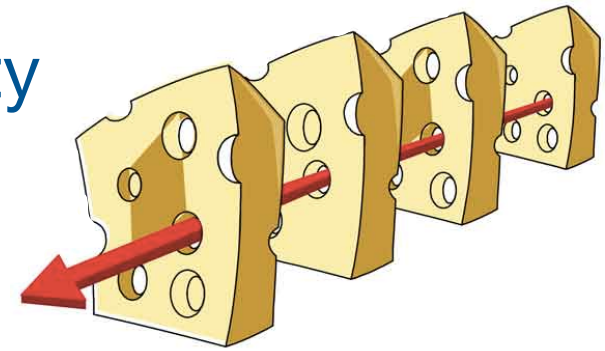
Principle: Never Rely Upon Obscurity

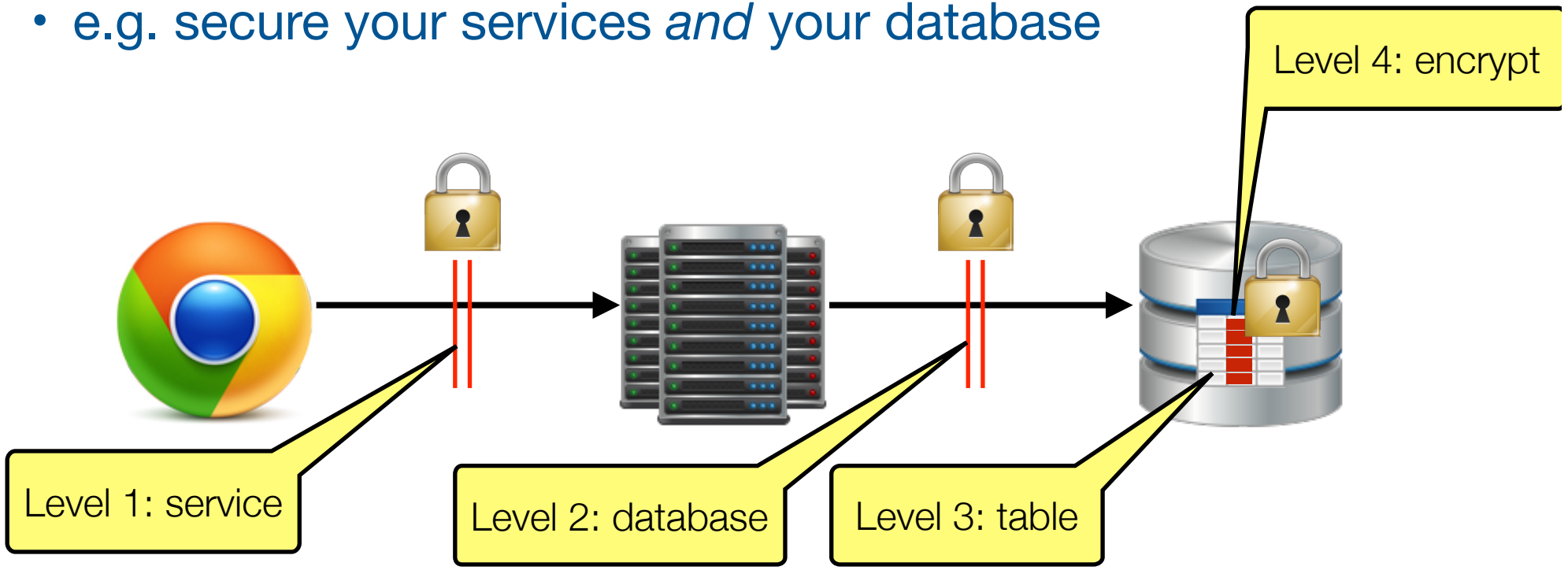


- Hiding secrets is difficult
  - someone is going to find them
  - accidentally it not on purpose
  - particularly susceptible to “fuzz” attacks
- Assuming people don’t figure things out is almost always wrong
  - a lot of attackers have time on their hands
  - they have to be lucky once
- The right approach is to assume that an attacker has perfect knowledge of your system
  - this forces secure system design

Principle: Defence in Depth

- Don’t rely on one level or type of security
  - secure at every level in your application
  - failures at one level should be stopped at the next
  - e.g. secure your services *and* your database





Summary

Summary

- We've looked how to improve system security
  - We've not talked about technology very much
  - as designers we need to be risk and principle driven
- Security requires: People, Process and Technology
  - the weakest of the three is how secure you are
- Security needs to be designed in
  - its very difficult and expensive to add later
- Be guided by risks not security technologies
  - threat risk models (STRIDE and DREAD); attack trees
- Get the experts involved for significant risks
  - and *never* invent your own security technology!

Summary (ii)

Never stop asking “why?” and “what if?”

critically important info sec questions!

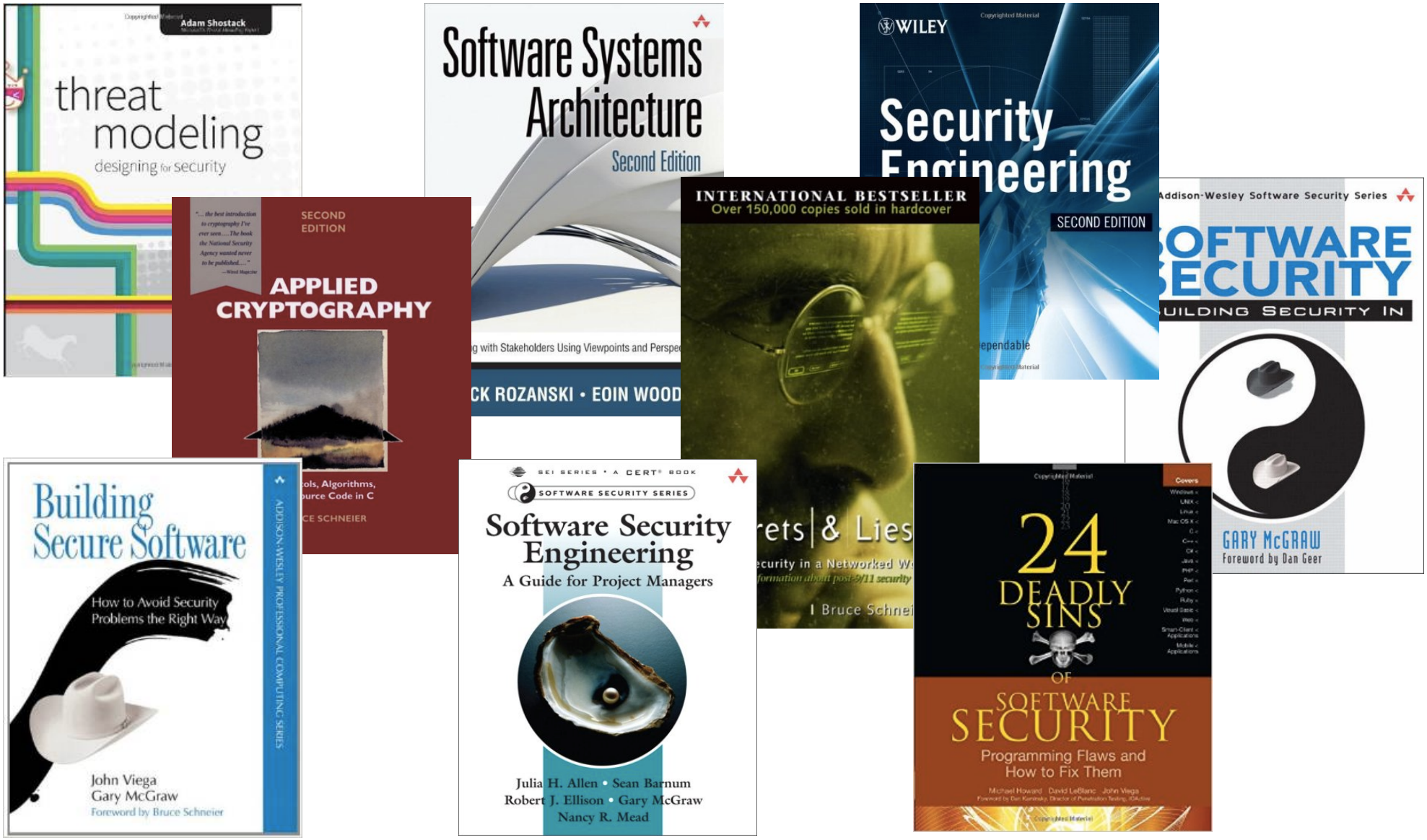
Resources

- OWASP - http://www.owasp.org
  - cookbooks, guides, sample apps, tutorials, …
- Microsoft SDL - http://www.microsoft.com/security/sdl
  - complete security development lifecycle with resources
- Elevation of Privilege game- http://tinyurl.com/eopgame
  - card game which helps to explain and drive threat modelling
- Trike- http://www.octotrike.org
  - alternative threat modelling approach
- CWE - http://cwe.mitre.org
  - vulnerability central

Resources

- CPNI - http://www.cpni.gov.uk
- US Government CERT - https://www.us-cert.gov
- CMU’s CERT - http://cert.org

Books


Adam Shostack
threat modeling
designing for security
Software Systems Architecture
Second Edition
WILEY
Security Engineering
SECOND EDITION
INTERNATIONAL BESTSELLER
Over 150,000 copies sold in hardcover
Addison-Wesley Software Security Series
SECOND EDITION
APPLIED CRYPTOGRAPHY
Building Secure Software
How to Avoid Security Problems the Right Way
John Viega
Gary McGraw
Foreword by Bruce Schneier
ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES
SEI SERIES • A CERT® BOOK
SOFTWARE SECURITY SERIES
Software Security Engineering
A Guide for Project Managers
Julia H. Allen • Sean Barnum
Robert J. Ellison • Gary McGraw
Nancy R. Mead
24 DEADLY SINS OF SOFTWARE SECURITY
Programming Flaws and How to Fix Them
GARY McGRAW
Foreword by Dan Geer

Questions?

Eoin Woods
www.eoinwoods.info
@eoinwoodz