UNIVERSITY OF CAMBRIDGE
Computer Laboratory

# Security II: Part 2: Security engineering

Daniel R. Thomas

Department of Computer Science and Technology, University of Cambridge

Lent 2018

GPG: 5017 A1EC 0B29 08E3 CF64 7CCD 5514 35D5 D749 33D9
https://www.cl.cam.ac.uk/~drt24/

This part of the course is completely independent from Part 1. I am a post-doctoral researcher in the Cambridge Cybercrime Centre. My research interests are in measuring security. In particular I am currently working on measuring DDoS from UDP reflection and botnets such as Mirai.

# Outline I

# Outline II

# Outline III

# Lecturers and lectures

- **Dr Daniel Thomas** (x5): Security, human factors, and psychology; security policies; authentication; network security.
- **Dr Richard Clayton** (x1): Security economics (14th February).
- **Dr Steven Murdoch** (x1): Anonymity and censorship resistance (19th February).
- **Dr Markus Kuhn** (x1): Web application security (23rd February).

# Supervisions and exams

## Supervisions

- 2 supervisions
- Material on the course website

## Exams

- One exam question set by Daniel Thomas on Part 2 of the course
- All material by all lecturers may be covered.

The supervision material linked from the course website is on Otter.
I thought of parts of question for all lectures on this part of the course.
The exam question should understanding not rote learning of details.

# Resources are available

- Ross Anderson's "Security Engineering" (2nd edition, 2008)
- Free: `https://www.cl.cam.ac.uk/~rja14/book.html`
- Further papers and resources cited as we go along

Ross Anderson's book is very readable and hasn't dated too badly, though there are some newer topics it does not cover. I would advocate reading the whole thing.

# Get involved

- Security seminars: Tuesdays LT2, 14:00-15:00 `https://talks.cam.ac.uk/show/index/5695`
- Security group meetings: Fridays, FW11, 16:00-17:00
- `https://lists.cam.ac.uk/mailman/listinfo/cl-security-research`

The security seminars are often very interesting and will help you go beyond the course. The group meetings often centre around discussion of topical issues but we also sometimes have various interesting visitors. Seven years ago I was sitting where you are sitting now being told this and I did get involved and as a result I am now standing here telling you the same thing.

# Format

- I will ask you questions as we go along **[Audience participation]**
  - Don't worry about giving the wrong answer
  - Chocolate
- If I say something unclear then stick your hand up
- Additional questions at the end or by email (drt24)
- New lecture notes

In order to try and keep you awake and make sure that you understand the material we are covering I am going to periodically ask you questions as that will help you concentrate and remember more of the lecture. There will be prizes for those who answer (regardless of whether the answer is correct).

If I say something unclear then please stick your hand up so I can clarify. If you have additional questions then catch me at the end or send me an email and I will repeat to the whole class if I something important comes up.

This is a new set of lecture notes, some sections are based off slides from Ross Anderson or Frank Stajano but there will be mistakes. They include the slides and in some cases additional text of some of the things I will say for that slide. Please let me know when you spot mistakes so that I can correct them. Please also fill in the course feedback a the end of the term so that I can learn how to improve.

# Introduction

# Aim: Make you a security engineer

Teach security engineering as a systems discipline

- What should be protected?
- How can it be protected?
- How do these systems interact such that the resultant behaviour is secure or insecure?

# Security engineering is about systems

Security engineering is about building **systems** to remain dependable in the face of malice, error, and mischance.

As a discipline, it focuses on the tools, processes, and methods needed to design, implement, and test complete **systems**, and to adapt existing **systems** as their environment evolves.

# Systems are broad

- A **system** can be
  - a product or component (PC, smartcard, ...)
  - some products plus OS, comms and infrastructure
  - + applications
  - + internal staff
  - + customers / external users
  - + regulatory environment
  - + economy
  - + planet
- Common failing: policy drawn to narrowly
  - For a secure system we need to consider **people**.

It is easy for computer scientists to forget the human component of security, and so that is where we are going to start and stay for the next few lectures.

# Security, human factors, and psychology

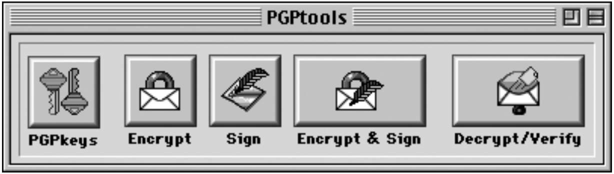
We need to understand how people work and what they understand so that we can build systems that actually work with real people.

People do not need to understand the mechanisms providing security as long as those mechanisms do not rely on being understood.

# Why Johnny can't encrypt[1]

Don't (implicitly) require users to have studied cryptography to use your product.

- 90% of encryption program PGP's users couldn't use it correctly within 90 minutes.
- Complex concepts: private/public, encryption/signing, RSA/DSA



[1] Alma Whitten and J.D. Tygar. 1999. Why Johnny Can't Encrypt. In *USENIX Security Symposium*. (Aug. 1999), 679–702. http://www.doug-tygar.com/papers/Why_Johnny_Cant_Encrypt/OReilly.pdf.

First paper to look at security usability. They did a user study with experienced email users with a role playing scenario or running a political campaign. Emails were sent unencrypted, private keys were sent and users generated keys for the people they were trying to communicate with. Subtle incompatibilities between RSA and DSA keys were particularly hard to understand.

# Users are not the enemy[2]

- Insufficient communication with users produces unusable systems
- Users forced to comply with password mechanisms incompatible with work practices will look for workarounds
- Vicious circle:
  - Security departments think users are inherently insecure
  - Users think security departments get in the way of real work
- Users are motivated to behave securely
- Treat users as stakeholders explain the real risks
- Provide feedback, guidance, awareness; and usable security

When were you treated as the enemy? **[Audience participation]**

[2] Anne Adams and Martina Angela Sasse. 1999. Users are not the enemy. *Communications of the ACM*, 42, 12, 40–46. ACM.

# Manage the compliance budget[3]

- Users assess cost/benefits of security measures and put up with some inconvenience for the good of the company
  - With virus scanner on, program takes longer to build
  - Encrypting USB stick might prevent me giving talk
- But only up to a point!
  - User patience is finite: “the compliance budget”
  - Once exhausted, the user stops cooperating

*Must be managed like any other budget*

[3] Adam Beautement, M. Angela Sasse, and Mike Wonham. 2008. The compliance budget: Managing security behaviour in organisations. *Proceedings of the New Security Paradigms Workshop (NSPW)*, 47–58. ACM.

# Think like an attacker

- Kevin D. Mitnick[4]
  - You don't have to pick the lock or break into the server: get someone on the inside to open the door for you
  - Pretext calls surprisingly effective
  - Skim read: verbose and repetitive writing.
- Traditional responses
  - mandatory access control
  - operational security

*But why do the attacks work?*

[4] Kevin D. Mitnick and William L. Simon. 2002. *The art of deception: Controlling the human element of security*. Wiley. ISBN: 978076454280, Kevin D. Mitnick and William L. Simon. 2005. *The art of intrusion: The real stories behind the exploits of hackers, intruders & deceivers*. Wiley. ISBN: 9780764569593.

# Understanding scam victims I

Seven principles for systems security[5]

### Distraction

While you are distracted hustlers can do anything and you won't notice.

### Dishonesty

Your larceny is what hooks you. Thereafter, anything illegal you do will be used against you by the fraudster.

Distraction: Phishing: user is fixated on task completion e.g finding why now payee on PayPal account

Dishonesty: Stolen goods: you try to buy cheap obviously stolen goods and now the fraudster takes your money and blackmails you not to report it.

Kindness: Claim to need someone's help to change 'your' tyre and then borrow their keys to warm up in their car, drive their car away. It wasn't your car you were trying to change the tyre on anyway.

Need and greed: Hustlers know you want a job as a lorry driver so they send you an application form for a lorry driver job that asks for all your personal details and then use that for identity theft.

Social compliance: Classic "Health and Safety Inspector" trick used in films and TV.

Herd: Get someone to play a shell game by having several conspirators playing it and seeming to make money but if you play then you always lose because your fellow players are conspiring.

Time: "We will close your email account in 30 minutes unless you do this." Taking a moment to give yourself time to think can help you make better choices.

# Understanding scam victims II

### Kindness

People are willing to help. Hustlers shamelessly take advantage.

### Need and greed

Once hustlers know what you want they can easily manipulate you.

### Social compliance

Society trains people not to question authority. Hustlers exploit this “suspension of suspiciousness” to manipulate you.

# Understanding scam victims III

## Herd

People let their guard down when others next to them appear to share the same risk. No safety in numbers if they are against you.

## Time

When under time pressure people use instincts and heuristics rather than logic. Hustlers use this to trick you.

[5] Frank Stajano and Paul Wilson. 2011. Understanding scam victims. *Communications of the ACM*, 54, 3, 70–75. ACM.

# Jewelry-shop scam[6]

Jess attempts to buy an expensive necklace but is “arrested” by Alex and Paul posing as plainclothes police officers who expose her as a well-known fraudster, notorious for paying with counterfeit cash. The “cops” collect as evidence the “counterfeit” (actually genuine) cash and, crucially, the necklace, which will, of course, “be returned.” The jeweler is extremely grateful the cops saved her from the evil fraudster.

What principles are being used here? **[Audience participation]**

[6] Frank Stajano and Paul Wilson. 2011. Understanding scam victims. *Communications of the ACM*, 54, 3, 70–75. ACM.

# Seven principles for systems security

- Principles for persuasion in general (there are other similar sets of principles)
- Based on human nature
- You have to accept human nature like the laws of physics
- You must design systems that work with real humans

# Weapons of influence[7]

Based on undercover study of salespeople “and other compliance professionals”

- Reciprocation: they will feel compelled to respond
- Commitment and consistency: “but you previously said X”
- Social proof: like to do what others do
- Liking: want to deal with people they can relate to
- Authority: will defer to authority figure
- Scarcity: less is best and loss is worst

[7] Robert B. Cialdini. 2014. *Influence: science and practice*. (5th ed.). Pearson Education Limited, Harlow, Essex. ISBN: 9781292035499.

Salespeople are like scam artists, except hopefully legal.

Reciprocation: “Would you like some tea?” Public sector workers are often bound to say no or fill in a form to disclose the gift which attempts to mitigate against this. Off-book dinners organised by lobbyists for people working at regulators can have a powerful influence.

Commitment: “You said you wanted a big garden...”

Social proof: “Professor Anderson bought one yesterday”

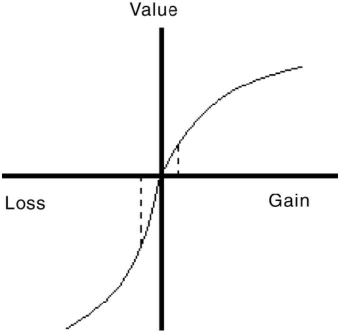
Liking: “My cousin went to your school”

Authority: “This is our laptop expert, she will help you choose the one that is right for you”

Scarcity: “Sale must end Tuesday”

# Bounded rationality

- Economics, Law, and Ethics course last year.
- Framing and risk aversion
- Prospect theory



Replace plot with vector graphic

Bounded rationality was covered in the Economics, Law, and Ethics course last year and so I will only briefly remind you of what you learnt on that course.

Framing effects include “Was £8.99 now £6.99” and estate agents showing bad property first.

Prospect theory: gains and losses matter, not overall wealth.

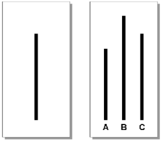
Both curves lean towards the horizontal, both higher gains and higher losses are progressively less relevant.

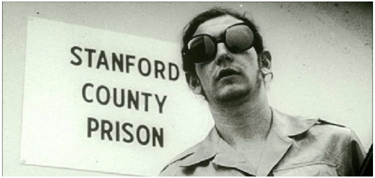
Steeper for losses than for gains, we dislike a loss much more than we like a win; and by about 2x.

Concave for gains (risk-averse), convex for losses (risk-seeking).

# Social psychology: Theory

- Solomon Asch, 1951 (conformity experiments). 2/3 of subjects deny obvious facts to conform to group
- Stanley Milgram, 1964: a similar number will administer torture if instructed by authority figure
- Philip Zimbardo, 1971 (Stanford prison experiment). The subjects' situation or context is enough





People will agree with the group that the answer is B even though it is clearly C.

Shows the Herd and Authority principles.

Not all these experiments could be repeated exactly under current ethical norms.

Giving people the roles of prisoners and prison wardens can lead to horrible behaviour, but there is some controversy around the lead researcher playing the prison superintendent and participants not being allowed to withdraw from the study. The results may not generalise.

# Social psychology: Practice

The Officer Scott case: a “police officer” phones a fast food restaurant and persuades the manager to strip-search and sexually humiliate an employee.

The manager has committed an offence and so has the caller, but the caller is harder to find.

# Phishing

- Tricking people into disclosing credentials by email
- By 2006, UK banks lost £35m and US banks $200m
- Early phish crude and greedy; but phishermen learn
- ‘Thank you for adding a new email address to your PayPal account’
- The banks make it easy for them: good phish are indistinguishable from the bank’s marketing emails

# Types of phishing website

- Misleading domain name
  `http://www.banckname.com`
  `http://www.bankname.xtrasecuresite.com`
- Compromised user website
  `http://www.example.com/~user/www.bankname.com/`
- Compromised machine
  `http://www.example.com/bankname/login/`
  `http://192.168.23.45/bankname/login/`
- Free web hosting
  `http://www.bank.com.freespacesitename.com`

## Fraud and phishing patterns

- Fraudsters do everything normal marketers do
- People learn by doing and marketers encourage clicking on (link tracking) links
- Banks blame the user and give advice they know can't be followed
- 'Look for the lock' or 'parse the URL' instructions only work for ICT professionals who know how HTTPS/HTML works.

Even banks own security teams can't distinguish marketing websites from phish. Bank's anti-phishing teams have got their own marketing websites taken down.

# Don't phish your own employees[8]

- No security benefit and harms employee productivity
- Even with regular training of staff ~20% will still fall for a good phish
- Don't allow phish to be delivered
- Assume people will fall for phish and build systems that are robust to this

[8] Steven J. Murdoch and Angela Sasse. 2017. Should you phish your own employees? (Aug. 22, 2017). Retrieved Jan. 4, 2018 from https://www.benthamsgaze.org/2017/08/22/should-you-phish-your-own-employees/.

People find phish in their spam folder and click on it (and complain to customer support that this important email went to spam). Build a system to detect real marketing emails using other techniques and then use the real marketing emails from banks to train your phishing classifier.

# Measuring the cost of cybercrime[9]

- Traditional offences low hundreds of £/€/$s per person per year
- Transitional frauds few £/€/$s per person per year
- New computer crimes 10s pence/cents per person per year
- Cost of fighting against computer crime an order of magnitude higher

[9] Ross Anderson, Chris Barton, Rainer Böhme, Richard Clayton, Michel J. G. van Eeten, Michael Levi, Tyler Moore, and Stefan Savage. 2012. Measuring the cost of cybercrime. In *Workshop on the Economics of Information Security*. Springer, Berlin, Germany, 265–300. ISBN: 978-3-642-39497-3.

Traditional offences: Tax evasion and welfare fraud have gone online
Transitional frauds: Credit card fraud has substantially changed its nature
New crimes: Denial of service and hacking
New crimes are like metal theft: they cost much more to deal with than the money the criminals make.

# Phishing losses

- Phishers earning up to $320 million per year in 2007[10]
- Most losses from banks are from man-in-the-browser trojans rather than phishing
- Recovered funds and losses blamed on customer are not counted
- UK: "Financial fraud losses across payment cards, remote banking and cheques totalled £768.8 million in 2016, an increase of 2 per cent compared to 2015." £6.40 in every £10 of fraud is stopped.

[10] Tyler Moore and Richard Clayton. 2007. Examining the impact of website take-down on phishing. In *APWG eCrime Researchers Summit*. ACM, 1–13. ISBN: 9781595939398.

Customer blaming is discriminatoryRoss Anderson. 2017. Banks biased against black fraud victims. (Jan. 2017). Retrieved Feb. 8, 2018 from https://www.lightbluetouchpaper.org/2017/01/12/banks-biased-against-black-fraud-victims/

Recent information on payment fraud is available here:
https://www.financialfraudaction.org.uk/fraudfacts17/

# Security policies

What are security policies?
Why do we have security policies?
Some examples of early security policies.

# Managing security

- Security awareness: measures must have, and be seen to have, full support of management
- Measuring security is hard
  - Measure: security vulnerabilities, attack surface, attack cost...
- Risk analysis
  - Assets, vulnerabilities, threats, probabilities
  - Inputs are guesswork
- Security policy: an instrument of communication

# Design hierarchy

| | |
|---|---|
| Policy | What are we trying to do? |
| How? | Protocols, procedures, ... |
| With what? | Hardware, software, crypto, ... |

# Terminology: Trust

*Trust* is hard to define accurately

1. A warm fuzzy feeling
2. A trusted system or component is one that can break my security policy
3. A trusted system is one I can insure
4. A trusted system won't get me fired when it breaks

In general we use the NSA definition (2).
E.g. an NSA employee selling key material to the Chinese is *trusted* but *not trustworthy*.

# Terminology

Security policy: a succinct statement of protection goals – typically less than a page of normal language.

Protection profile: a detailed statement of protection goals – dozens of pages of semi-formal language

Security target: a detailed statement of protection goals applied to a particular system including both functionality and testing.

# Bad policies

1. This policy is approved by Management
2. All staff shall obey this security policy
3. Data shall be available only to those with a ‘need to know’.
4. All breaches of this policy shall be reported at once to Security.

What is wrong with this? **[Audience participation]**

# Policy: Multi-level security

- Multilevel Secure (MLS) systems widely used in government contexts.
- With ‘Secret’ clearance can read ‘Official’ and ‘Secret’ but not ‘Top secret’
- First security policy to be worked out in detail, recommended keeping security policy and enforcement simple.[11]

[11] James P. Anderson. 1972. Computer security technology planning study. Tech. rep. Electronic Systems Division, Air Force Systems Command, Hanscom Field, Bedford, MA, (Oct. 1972). `http://seclab.cs.ucdavis.edu/projects/history/CD-1/ande72.pdf`.

# Levels of information[12]

Top Secret: Most sensitive information. Compromise could cause widespread loss of life or threaten security or economic wellbeing of the country.

Secret: Very sensitive justifying heightened protections. Compromise could damage military capability, international relations or aid serious organised crime.

Official: Majority of information created or processed by public sector. Could be damaging consequences to compromise.

[12] Cabinet Office. 2013. Government Security Classifications. Tech. rep. (Oct. 2013). https://www.gov.uk/government/publications/government-security-classifications.

# Application of MLS

- Resources have classifications
- Principles have clearances
- Information flows upwards only

# Context of MLS

- Information must not leak from High to Low
- Enforcement must be independent of user actions
- Perpetual problem: careless staff
- 1970s worry: operating system security
- 1990s worry: virus at Low copies itself to High and starts signalling down (e.g. covert channel)

Manning (2010) and Snowden (2013) show us how things actually go wrong in practice...

Manning: it is all in one big pot with lots of people who can access it and people in field offices can walk out with CDs full of top secret content.
Snowden: System administrators can gain additional privileges and use them to access more than they are supposed to have access to and know enough opsec to get away with it.

# Practice of MLS[13]

- September 2009: Dalai Lama's office realised there had been a security failure
- Initial break: Targeted email with bad pdf
- Then: Took over the mail sever and spread it
- About 35 of their 50 PCs were infected
- Fix (Dharmsala): take 'Secret' stuff offline
- Fix (UKUSA agencies): use MLS mail guards and firewalls to prevent 'Secret' stuff getting out

[13] Shishir Nagaraja and Ross Anderson. 2009. The snooping dragon: social-malware surveillance of the Tibetan movement. Tech. rep. 746. Computer Laboratory, (Mar. 2009), 1–12. https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-746.html.

# Bell-LaPadula (BLP) Policy[14]

- Simple security (ss)-property: no read up
- *-property: no write down
- With these, one can prove that a system which starts in a secure state will remain in one
- Ideal: Minimise the Trusted Computing Base so its verifiable
- 1970s idea: use a reference monitor

[14] D.E. Bell and L.J. LaPadula. 1973. Secure computer systems: Mathematical Foundations. Tech. rep. Electronic Systems Division, Air Force Systems Command, United States Air Force, L.G. Hanscom Field, Bedford, MA, (Nov. 1973). `http://www.dtic.mil/docs/citations/AD0770768`.

In reality proving that a system starts in a secure state is hard and in practice mistakes or flaws will cause the state to transition to insecure. In the model there is no way back to secure from insecure because it is supposed to be impossible.

The Trusted Computing Base (TCB) is the set of hardware, software, and procedures that can break the security policy.

# The lattice model I

- Intelligence agencies manage ‘compartmented’ data by adding categories. `Label = (level, {set of categories})`
- BLP require only a partial order (*dominates*)
- $X$ dominates $Y$ iff $\mathrm{level}(X) \geq \mathrm{level}(Y) \wedge \mathrm{cat}(X) \supseteq \mathrm{cat}(Y)$

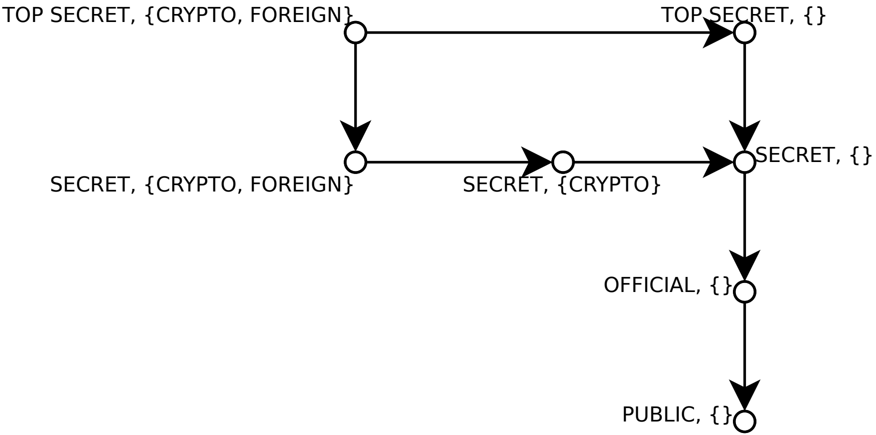
In the diagram transitive edges are not shown.

Note that “TOP SECRET, $\{\}$” does not dominate “SECRET, $\{$CRYPTO$\}$” as “dominates” is only a partial order.

Consider Levels as one dimension with four states and each category is binary dimension and then dominates is $\geq$ in that space.

No compartments at OFFICIAL or PUBLIC as the security mechanisms in use at those levels can’t support them.

# The lattice model II



# The lattice model III

- BLP simple property (NRU): X can read Y iff X dominates Y
- BLP *property (NWD) X can write Y iff X is dominated by Y

# Objections to BLP: TCB growth

- Some processes such as memory management, need to read and write at all levels.
- Fix: put them in the trusted computing base
- Consequence: once you put all the stuff a real system needs (backup, recovery, comms...) the TCB is no longer small enough to be easily verifiable

However, there are tricks with microkernels (seL4) and hardware capabilities (CHERI) that can mitigate this TCB growth.

seL4: Gerwin Klein, Kevin Elphinstone, Gernot Heiser, June Andronick, David Cock, Philip Derrin, Dhammika Elkaduwe, Kai Engelhardt, Rafal Kolanski, Michael Norrish, Thomas Sewell, Harvey Tuch, and Simon Winwood. 2009. seL4: Formal verification of an OS kernel. In *Proceedings of the ACM SIGOPS 22nd Symposium on Operating System Principles*. ACM, Big Sky, Montana, USA, (Oct. 2009), 207–220. ISBN: 9781605587523

CHERI: Robert N M Watson, Jonathan Woodruff, Peter G. Neumann, Simon W. Moore, Jonathan Anderson, David Chisnall, Nirav Dave, Brooks Davis, Khilan Gudka, Ben Laurie, Steven J. Murdoch, Robert Norton, Michael Roe, Stacey Son, and Munraj Vadera. 2015. CHERI: A hybrid capability-system architecture for scalable software compartmentalization. *Symposium on Security and Privacy*, 20–37. IEEE. ISSN: 10816011

# Objections to BLP: Flow to High

- Usually a process acquires the highest label of any resource it has touched. So eventually everything is TOP SECRET.
- Applications have to be re-written to cope with that.

This can cause problems for things like licence servers as you need one for each level for software at that level to use and then what do you do about applications which change levels and now need to talk to a different licence server.

An active downgrading process is required to stop everything ending up TOP SECRET and downgrading is hard (see later).

# Objections to BLP: Receipts

- High can't acknowledge receipt from Low
- Information vanishes into a black hole
- Option 1: Accept this and engineer for it – CIA usenet feed
- Option 2: Allow acks but be aware they might be used by High to signal to Low. Address with some combination of software trust and covert channel elimination.

Usenet was a message board/forum/mailing list like system from before the web where people would post messages and discuss things. The CIA wanted to monitor Usenet as it contained lots of interesting public discussions but they could not safely connect a machine with classified information to Usenet in case classified information got posted to Usenet. So they built a 1-way pump for Usenet data with the classified systems on one side and the public Internet on the other.

By verifying the software and eliminating covert channels (fixed disk usage, memory, and CPU allocations for each level) risk can be reduced.

# Covert channels

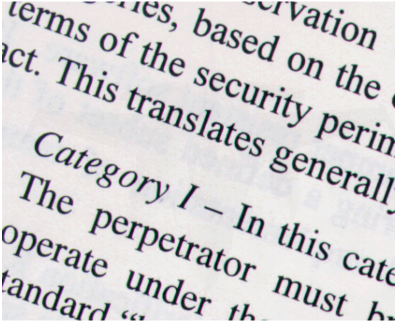
- A Trojan at High signals to a buddy at Low by modulating a shared system resource: e.g. disk space or CPU load.
- Capacity depends on bandwidth and S/N. So cut bandwidth or increase noise.
- Really hard to get below 1 bit/s

This makes it very difficult to protect against the leakage of private keys or other small but valuable secrets. It can be good enough to prevent the leak of classified designs or high volume intelligence.

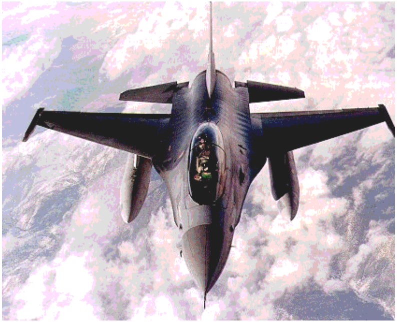
# Downgrading

- How can information be downgraded?
- Analysts produce SECRET briefings based on TOP SECRET intelligence, by manual paraphrasis
- Some objects are downgraded by a trusted subject such as satellite images

# Downgrading of images



ervation
es, based on the
terms of the security perin
ct. This translates generall
Category I – In this cate
The perpetrator must b
operate under th
tandard "



Picture hidden in three least significant bits of text

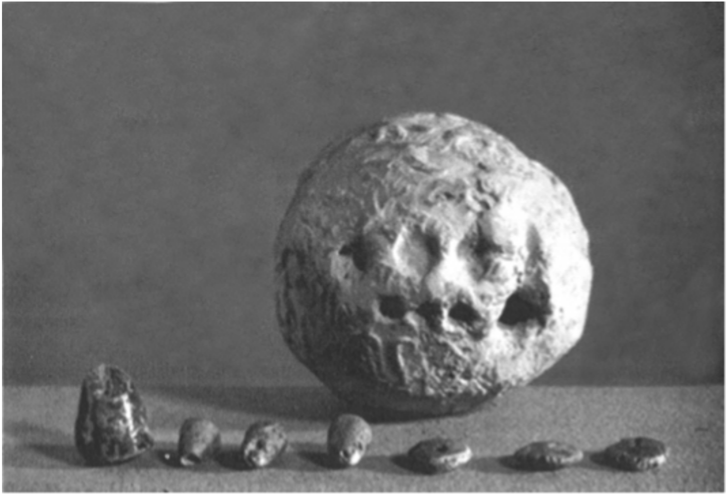
# How does the lattice model fail?

**[Audience participation]**

# Biba model: Multilevel Integrity

- Data may only flow down from high-integrity to low-integrity
- Dual of BLP:
  - Simple integrity property: subject may write to object iff object has same or lower label as subject
  - *-integrity property: subject may read object iff object has same or higher label as subject
- Provides low watermark properties, etc.
- Example: medical equipment with two levels "calibrate" and "operate".

# Bookkeeping, c. 3300 BC



This clay tablet and the associated tokens is an early accounting ledger or record of a transaction.

# Bookkeeping c. 1100 AD

- How do you manage a business that's become too large to staff with your own family members?
- Double-entry bookkeeping – each entry in one ledger is matched by opposite entry in another
  - E.g. firm sells £100 of goods on credit – credit the sales account, debit the receivables account
  - Customer pays – credit the receivables account, debit the cash account
- So bookkeepers have to collude to commit fraud

Credits and debits here may sound backwards but make sense to accountants. For a more computer science friendly explanation see: Martin Kleppmann. 2011. Accounting for Computer Scientists. (Mar. 2011). Retrieved Jan. 29, 2018 from http://martin.kleppmann.com/2011/03/07/accounting-for-computer-scientists.html, https://perma.cc/Z66K-YYJT

# Banking security policy

- Threat model:
  - 1% of staff go bad each year
  - Mistakes happen – 1 in 500 paper transactions
  - There are clever fraudsters too
  - Loss of confidence means ruin
- Protection goals:
  - Deter/prevent the obvious frauds
  - Detect the rest as soon as possible
  - Be able to defend the bank's actions in court

# Availability

- Availability can matter a lot more than integrity/authenticity or confidentiality in industry.
- Unavailable implies no income
- Availability is also important for burglar alarms etc.

How do you rob a jewellery store? All the stores alarm systems go back to the exchange and have tamper evident cables so the alarm goes of if the cables are damaged. Small explosion at the exchange sets off all the alarms. Police can't go to all the shops and so you break into one of them.

How do you break in to a secure facility? Destroy trust in the alarm system by triggering lots of "false positives" perhaps on a stormy night, eventually no one responds to the alarms. At that point cut the fence with impunity.

# Policy

- We have seen some examples of policies
- Many industries develop their own policies, and get Protection Profiles evaluated
- Many things go wrong – people protect the things they can, not the things they should
- We often see deception at the policy level!

# Developing security policies I

**"No worries, our product is 100% secure. All data is encrypted with 128-bit keys. It takes billions of years to break these."**

Such statements are abundant in marketing literature. A security manager should ask:

- What does the mechanism achieve?
- Do we need confidentiality, integrity or availability of exactly this data?
- Who will generate the keys and how?
- Who will store / have access to the keys?

# Developing security policies II

- Can we lose keys and with them data?
- Will it interfere with other security measures (backup, auditing, scanning, …)?
- Will it introduce new vulnerabilities or can it somehow be used against us?
- What if it breaks or is broken?
- …

# Security policy development: Step 1: Security requirements analysis

- Identify assets and their value
- Identify vulnerabilities, threats and risk priorities
- Identify legal and contractual requirements

Have you drawn your requirements widely enough?
Are there stakeholders you have missed out?

# Step 2: Work out a suitable security policy

The security requirements identified can be complex and may have to be abstracted first into a high-level **security policy**, a set of rules that clarifies what is or is not authorised, required and prohibited activities, states, and information flows.

**Security policy models** are techniques for the precise and even formal definition of such protection goals. They can describe both automatically enforced policies (e.g., a mandatory access control configuration in an operating system, a policy description language for a database management system, etc.) and procedures for employees (e.g., segregation of duties).

# Step 3: Security policy document

Once a good understanding exists of what exactly security means for an organisation and what needs to be protected or enforced, the high-level security policy should be documented as a reference for anyone involved in implementing controls. It should clearly lay out the overall objectives, principles and the underlying threat model that are to guide the choice of mechanisms in the next step.

# Step 4: Selection and implementation of controls I

Issues addressed in a typical low-level organisational security policy:

- General (affecting everyone) and specific responsibilities for security
- Names manager who "owns" the overall policy and is in charge of its continued enforcement, maintenance, review, and evaluation of effectiveness
- Names individual managers who "own" individual information assets and are responsible for their day-to-day security
- Reporting responsibilities for security incidents, vulnerabilities, software malfunctions
- Mechanisms for learning from incidents

Background checks, supervision, confidentiality agreement

Definition of security perimeters, locating facilities to minimise traffic across perimeters, alarmed fire doors, physical barriers that penetrate false floors/ceilings, entrance controls, handling of visitors and public access, visible identification, responsibility to challenge unescorted strangers, location of backup equipment at safe distance, prohibition of recording equipment, redundant power supplies, access to cabling, authorisation procedure for removal of property, clear desk/screen policy, etc.

Avoid that a single person can abuse authority without detection (e.g., different people must raise purchase order and confirm delivery of goods, croupier vs. cashier in casino)

What activities are logged, how are log files protected from manipulation

Zeroise, degauss, reformat, or shred and destroy storage media, paper, carbon paper, printer ribbons, etc. before discarding it.

# Step 4: Selection and implementation of controls II

- Incentives, disciplinary process, consequences of policy violations
- User training, documentation and revision of procedures
- Personnel security (depending on sensitivity of job)
- Regulation of third-party access
- Physical security
- Segregation of duties
- Audit trails
- Separation of development and operational facilities
- Protection against unauthorised and malicious software

# Step 4: Selection and implementation of controls III

- Organising backup and rehearsing restoration
- File/document access control, sensitivity labeling of documents and media
- Disposal of media
- Network and software configuration management
- Line and file encryption, authentication, key and password management
- Duress alarms, terminal timeouts, clock synchronisation, …

# Guidance on writing security policy documents

- British Standard 7799 “Code of practice for information security management”
- German Information Security Agency’s “IT Baseline Protection Manual”
  `http://www.bsi.bund.de/english/gshb/manual/`
- US DoD National Computer Security Center Rainbow Series, for military policy guidelines
  `http://en.wikipedia.org/wiki/Rainbow_Series`

# Authentication

Authentication: checking who a principal is.
Authorisation: checking that an authenticated principal is allowed access.

# Passwords are rubbish but dominate

- Usability problems
- Security problems
- Dominate authentication practice

“There is no doubt that over time, people are going to rely less and less on passwords. People use the same password on different systems, they write them down and they just don’t meet the challenge for anything your really want to secure.” (Bill Gates, keynote @ RSA conference, **2004**)

Passwords are hard to use as they rely on remembering random strings and people find it hard to remember which password they used for a particular service.

Phishing, offline and online bruteforcing can be used to break into systems reliant on password based authentication

Many schemes have been proposed to replace passwords but none have succeeded and password based authentication still dominates.

# Modern password policies are better

- Blacklist most common passwords
- Change passwords only on suspected compromise
- Prioritise administrator and remote user accounts
- Use secure storage for passwords
- Rate limit attempts
- Change default passwords

National Cyber Security Centre

# Password security

Attackers use a variety of techniques to discover passwords, including using powerful tools freely available on the internet. The following advice makes password security easier for your users – improving your system security as a result.

## How passwords are cracked...

### Interception

Passwords can be intercepted as they are transmitted over a network.

### Brute Force

Automated guessing of billions of passwords until the correct one is found.

### Searching

IT infrastructure can be searched for electronically stored password information.

### Stealing Passwords

Insecurely stored passwords can be stolen – this includes handwritten passwords hidden close to a device.

### Manual Guessing

Personal information, such as name and date of birth can be used to guess common passwords.

### Shoulder Surfing

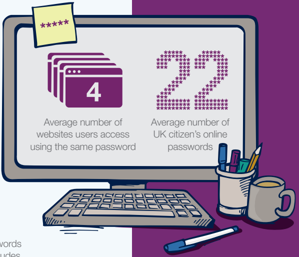
Observing someone typing their password.

### Social Engineering

Attackers use social engineering techniques to trick people into revealing passwords.

### Key Logging

An installed keylogger intercepts passwords as they are typed.



Average number of websites users access using the same password

Average number of UK citizen's online passwords

## ...and how to improve your system security

### Help users cope with 'password overload'

- Only use passwords where they are really needed.
- Use technical solutions to reduce the burden on users.
- Allow users to securely record and store their passwords.
- Only ask users to change their passwords on indication of suspicion of compromise.
- Allow users to reset password easily, quickly and cheaply.

### Help users generate appropriate passwords

- Put technical defences in place so that simpler passwords can be used.
- Steer users away from predictable passwords – and ban the most common.
- Encourage users to never re-use passwords between work and home.
- Train staff to help them avoid creating passwords that are easy to guess.
- Be aware of the limitations of password strength meters.

Blacklist the most common password choices

Monitor failed login attempts… train users to report suspicious activity

Prioritise administrator and remote user accounts

Don't store passwords in plain text format.

**** UPDATE

Change all default vendor supplied passwords before devices or software are deployed

Use account lockout, throttling or monitoring to help prevent brute force attacks

For more information go to **www.ncsc.gov.uk** **@ncsc**

# Password red flags

- Maximum password length (probably stored in plaintext)
- Certain characters forbidden (ditto)
- Password quoted in email (!)
- Password must be changed every 90 days

Complain to any organisation that exhibits these flaws

# Password verification has evolved[15]

1950s Store username and plaintext password [u,p]

1963 Store username and hash of password [u,h(p)]

1975ish Store username, salt and hash [u,s,h(s||p)]

1999 slower bcrypt hash with 128 bit salt

2009 scrypt hash which is harder to speed up in hardware

2015 Argon2id hash wins Password Hashing Competition

Today many systems still store the plaintext passwords which are then regularly leaked.

[15] Robert Morris and Ken Thompson. 1979. Password security: A case history. *Communications of the ACM*, 22, 11, 594–597. ACM.

Hashing of passwords first proposed at Cambridge by Roger Needham and Michael Guy in 1963 for the Titan system. Maurice V. Wilkes. 1968. *Time-sharing computer systems*. MacDonald & Co. isbn: 0356024261.

scrypt tries to thwart hardware based acceleration through using (relatively) large quantities of RAM which cannot then be included on the ASIC. Scrypt is parametrised.

Argon2id is parametrisable so that the quantity of work required to compute the hash can be varied as computers get faster and have more RAM.

It is important to store the version number of your password hashing scheme with each (salt, hash) pair so that you can change it later.

# We need salts to stop hash tables

- Without salts the same password always hashes to the same value
- You can tell if two users share the same password
- An offline attacker can hash all common passwords and check for matches
- Rainbow tables provide a time-space tradeoff that makes this tractable

# Offline vs online password strength

- Humans can learn passwords that resist online guessing
- Learning cryptographically strong passwords is hard (but possible[16])

Draconian password policies are just incompetent websites covering themselves[17] (but some don't realise this won't work with plaintext passwords).

[16] Joseph Bonneau and Stuart Schechter. 2014. Towards Reliable Storage of 56-bit Secrets in Human Memory. *USENIX Security Symposium (USENIX Security)* .

[17] Cormac Florencio, Dinei and Herley. 2010. Where do security policies come from? In *Symposium on Usable Privacy and Security (SOUPS)*. ACM, (July 2010) .

# Password database compromise is common[18]

- $>$40% of users reuse passwords
- $\sim$1% of sites had their passwords compromised within 1 year of measurement
- Half of those used plaintext password storage
- None of them notified their users that their passwords were compromised

Have you had your password compromised before? **[Audience participation]**

[18] Joe Deblasio, Stefan Savage, Geoffrey M. Voelker, and Alex C. Snoeren. 2017. Tripwire: Inferring Internet Site Compromise. In *Internet Measurement Conference (IMC)*. ACM, London, UK, (Nov. 2017) .

The authors automatically signed up to ∼2000 websites with an email address only used for that website and they used the same password on the email account. They signed up two addresses to each site, one with a weak guessable password and one with a cryptographically strong random password. They then monitored for logins on the email accounts. 100 000 control accounts did not get logged into. 1% of sites registered at did get logged into during the year of the study and of those half also had the strong password compromised indicating a plaintext leak.

This means your password has probably already been leaked and the organisation that leaked it will probably never tell you they were compromised, even if they knew.

# Single sign-on[19]

One single password to rule them all

- Doesn't even have to be a password
- Will it really rule "all"?

| | pseudo-SSO | true-SSO |
|---|---|---|
| local | | |
| proxy-based | | |

[19] Andreas Pashalidis and Chris J. Mitchell. 2003. A taxonomy of single sign-on systems. In *Australasian Conference on Information Security and Privacy (ACISP)*. vol. LNCS 2727. Springer, (July 2003), 249–264.

pseudo-SSO is where the user authenticates to the SSO component and then that component automatically complete the authentication with each service provider. The SSO component manages service provider specific credentials for each service provider.

In true-SSO the user authenticates to an Authentication Service Provider (ASP) and that provider has a relationship with all the service providers to authenticate the user to them. The service providers are explicitly trusting the ASP with true-SSO whereas with pseudo-SSO the user is explicitly trusting the pseudo-SSO component.

Local systems are resident within a device controlled by the user while proxy based ones use an external server.

# Local pseudo-SSO

Password managers

- Browser based
- Browser extensions
- Smartphone app based

What password managers do you use? **[Audience participation]**

# Local true-SSO and Proxy-based pseudo-SSO

These don't really exist as widely deployed systems.

However, Monkeysphere is a niche local true-SSO system based on GPG and SSH.

I also designed a totally unimplemented local true-SSO system.[20]

[20] Daniel R. Thomas and Alastair R. Beresford. 2014. Better authentication: Password revolution by evolution. In *Security Protocols XXII*. Bruce Christianson, James Malcolm, Vashek Matyáš, Petr Švenda, Frank Stajano, and Jonathan Anderson, (Eds.) Vol. LNCS 8809. Springer, Cambridge, UK, (Mar. 2014), 130–145. ISBN: 978-3-319-12399-8.

# Monkeysphere solves SSH public-key distribution

- Use GPG subkeys as SSH public keys
- Distribute validity information and revocation using normal key servers
- Killer application for GPG? In the DTG yes. Not widely deployed.

Developed by some Debian people – they already need to have GPG keys that are well maintained and also had servers they wanted to authenticate to. Managing SSH public-keys is tedious so Monkeysphere automates this.

This is the killer application for GPG keys in the DTG as if people don't use it then then can't authenticate to our servers. Usability is poor and it consumes a lot of compliance budget and so many users find workarounds. I like it though.

I will discuss SSH later on Slide 95.

# Proxy-based true-SSO

OpenID/OAuth Protocols used by ‘Sign-in with Google/Twitter’ (or DIY)

Facebook connect Protocol used by ‘Sign-in with Facebook’

Raven Ucam-Webauth protocol used by University of Cambridge

Shibboleth/SAML Federated SSO used by many Universities (backed by Raven in Cambridge)

Kerberos Network authentication system used by Windows Active Directory and Linux LDAP

Markus will discuss some of these in more detail later.

Experience with the isaacphysics.org project is that school children don't want to use Facebook or Google login, partly because they don't understand that using these services for authentication does not give the website they authenticated to access to their personal data. This is a reasonable concern, because if the APIs are used differently then such access can be requested and users might reasonably believe themselves unable to tell the difference.

# Passwords are here to stay[21]

- Cheapest for implementers
  - No need to explain them to users
  - Cost to support 1 more user is negligible
  - Can't be that bad because everyone uses them
- Many alternatives have been proposed
- While they might provide better security or usability they don't provide better deployability than the incumbent

Local pseudo-SSO reduces pain and proxy-based true-SSO is the best way forward for most sites.

[21] Joseph Bonneau, Cormac Herley, Paul C. van Oorschot, and Frank Stajano. 2012. The quest to replace passwords: A framework for comparative evaluation of web authentication schemes. In *IEEE Symposium on Security and Privacy* .

Facebook reached 1 million users before external funding, adding another user must have negligible cost.

Being easier to deploy than an incumbent technology is unusual.

# Second factor authentication: 2FA

Can substantially improve security by making it so that just having the password is not enough.

- Something you have forgotten
- Something you have lost
- Something you were

Recovery mechanisms are important.

The password you forgot.
The hardware token you have lost.
The fingerprints you had before you burnt your hand.
Something you have could be a device you have logged in from before or an authenticator app or the ability to receive text messages. People can't always receive text messages in the locations that they need to authenticate (poor coverage, basements etc.).

# One time password authentication

Second factor of a one time password generated using key material shared between client and server.

Implemented by Google Authenticator and many other products and supported by many services (Google, Github, Gitlab, Facebook, Twitter).

# HMAC-based One-time Password Algorithm (HOTP)

$S_K$ shared secret key

C shared counter

$\mathrm{HMAC}(K, m)$ $= \mathrm{SHA1}(K \oplus 0x5c5c \ldots || \mathrm{SHA1}(K \oplus 0x3636 \ldots || m))$

$\mathrm{Truncate}(x)$ Selects 4 bytes from $x$

$d$ token length (6)

$$\mathrm{HOTP}(S_K, C) = \mathrm{Truncate}(\mathrm{HMAC}(S_K, C)) \& 0x7FFFFFFF$$

$$\mathrm{HOTPValue}(S_K, C) = \mathrm{HOTP}(S_K, C) \mod 10^d$$

# Time-based One-time Password Algorithm (TOTP)

$T_C$ current time

$T_0$ epoch time (Unix epoch i.e. 1970-01-01)

$T_I$ Time interval granularity (30 seconds)

$S_K$ Shared secret key

$d$ token length (6)

$$T_C = \mathrm{floor}\left(\frac{\mathrm{unixtime(now)} - \mathrm{unixtime}(T_0)}{T_I}\right)$$

$$\mathrm{TOTP} = \mathrm{HOTP}(S_K, T_C) \quad \mathsf{mod}\ 10^d$$

# Biometric authentication

- Easier to do securely in contexts where you control the physical environment.
  - Authentication to local devices
  - Authentication at guarded checkpoints
  - Hard to use for remote authentication (can try stylometry etc.)
- Fingerprints
- Iris recognition and retina scans
- Facial recognition
- Voice recognition
- Stylometry, hand geometry

False positives and false negatives under normal, adverse, and malicious conditions.

If you use fingerprint authentication do thieves start removing fingers as well as stealing the car? With fingerprint recognition multiple fingerprints can be added for the same user, potentially for multiple people which allows safe account sharing, but on iOS at least facial recognition doesn't support multiple faces for one account.

Biometrics can't be revoked and can often be forged in unattended scenarios (e.g. with a photo).

# When you have to use passwords[22]

- Never receive the plaintext password (hash client side and then again on the server)
- Use rate-limiting
- Use strong cookies
- Do everything over perfectly forward secret TLS

Sign up to HaveIBeenPwned: `https://haveibeenpwned.com/`

[22] Joseph Bonneau. 2011. Getting Web Authentication Right A Best-Case Protocol for the Remaining Life of Passwords. *Security Protocols XIX*, 7114, 98–104. Springer.

Sign up your individual email accounts to HaveIBeenPwned and if you are responsible for a domain sign that up too. Then you will get notified when a leak of your password becomes public or when the leak of someone with an email in your domain becomes public and you can take appropriate action.

# Authentication is machine learning[23]

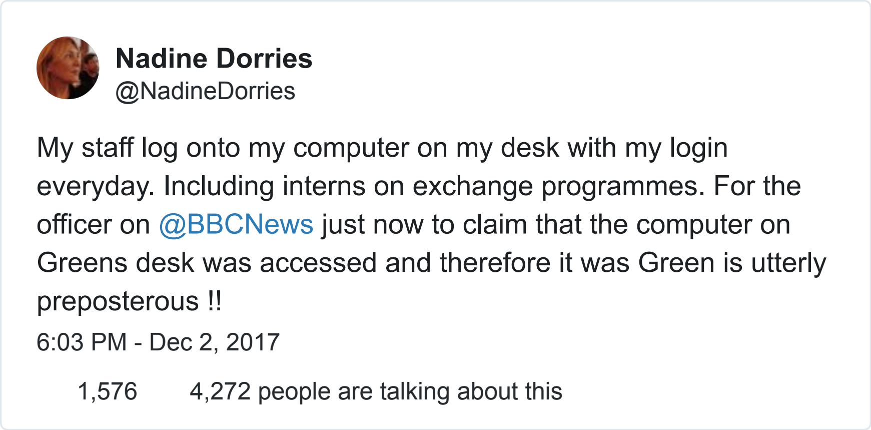
- If they are using the same device as yesterday on the same IP in the same way it is probably them even if they don't know the password
- If they know the password but are coming from a new device in a new country in a different way then the password is not enough
- Advantage to big players with lots of data

[23] Joseph Bonneau. 2012. Authentication is machine learning. (Dec. 2012). Retrieved June 26, 2013 from http://www.lightbluetouchpaper.org/2012/12/14/authentication-is-machine-learning/.

# Progressive authentication[24]

- The type of `authenticated` is `real` not `bool`
- Authenticated enough to read bank balance? Authenticated enough to send money?
- Proportionate authentication for the task at hand

[24] Oriana Riva, Chuan Qin, Karin Strauss, and Dimitrios Lymberopoulos. 2012. Progressive authentication: Deciding when to authenticate on mobile phones. *USENIX Security Symposium*. USENIX.

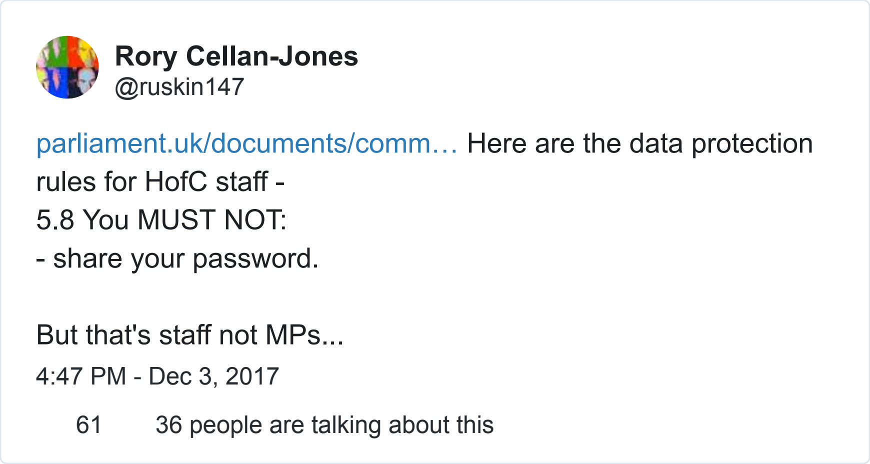


**Nadine Dorries**
@NadineDorries

My staff log onto my computer on my desk with my login everyday. Including interns on exchange programmes. For the officer on @BBCNews just now to claim that the computer on Greens desk was accessed and therefore it was Green is utterly preposterous !!

6:03 PM - Dec 2, 2017

1,576 4,272 people are talking about this



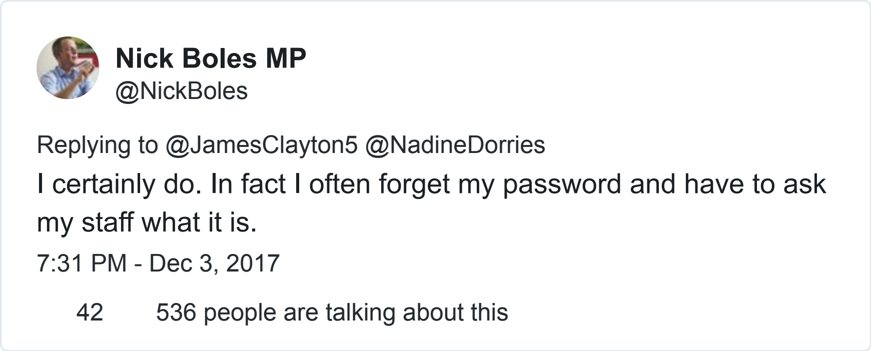
**Rory Cellan-Jones**
@ruskin147

parliament.uk/documents/comm… Here are the data protection rules for HofC staff -
5.8 You MUST NOT:
- share your password.

But that's staff not MPs...

4:47 PM - Dec 3, 2017

61 36 people are talking about this



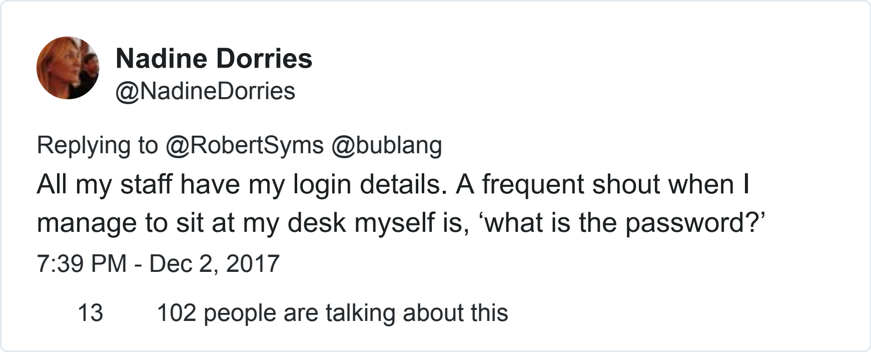
**Nick Boles MP**
@NickBoles

Replying to @JamesClayton5 @NadineDorries

I certainly do. In fact I often forget my password and have to ask my staff what it is.

7:31 PM - Dec 3, 2017

42 536 people are talking about this



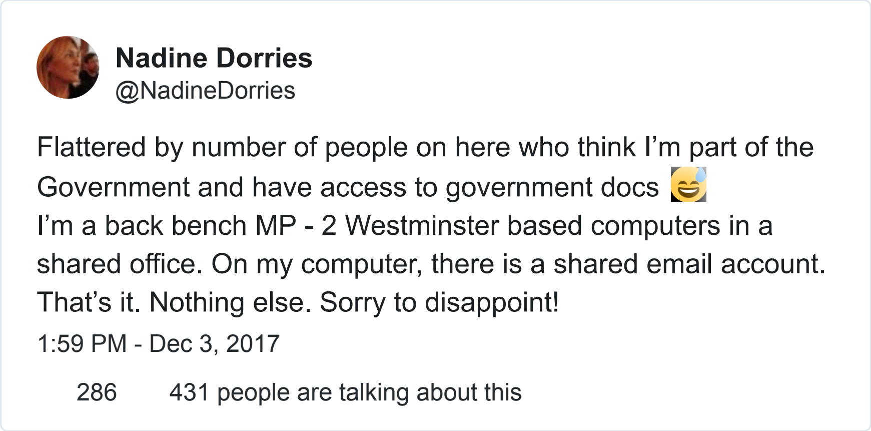
Nadine Dorries
@NadineDorries

Replying to @RobertSyms @bublang

All my staff have my login details. A frequent shout when I manage to sit at my desk myself is, ‘what is the password?’

7:39 PM - Dec 2, 2017

13 102 people are talking about this



**Nadine Dorries**
@NadineDorries

Flattered by number of people on here who think I'm part of the Government and have access to government docs 😅
I'm a back bench MP - 2 Westminster based computers in a shared office. On my computer, there is a shared email account. That's it. Nothing else. Sorry to disappoint!

1:59 PM - Dec 3, 2017

286 431 people are talking about this

Good or bad practice? **[Audience participation]**

# Network security

Brief tour through key topics in network security considering various technologies, attacks and defences.

# Take care: ethical and legal issues

Jurisdictions: Activity on networks can rapidly touch many countries: laws from all may apply.

Computer misuse: Unauthorised use of other people's computers or networks

Data protection: Metadata and content may contain PII

Scale: Networks can affect many people quickly so care is required

Visibility: Activity might be both invisible to some people but blatant to others.

# Firewalls

- Network and host based
- Stateless or statefull
- Application proxy (e.g. Web application firewall)
- Firewall vs NAT
- Universal Plug and Play (UPnP)
- Port forwarding

Both network wide and host based firewalls should be used. Network wide firewalls let you see what can be reached across your whole network. Host based ones let you impose more fine grained controls. For servers you want an explicit host firewall so that installing a new program doesn't result in a new publicly accessible service unless you meant it to (easy to manage with configuration management).

Stateless firewalls are much more efficient and statefull ones can fall over under high load, they need to maintain a connection table with a record per connection. However, stateless firewalls can't distinguish between incoming and outgoing connections. Statefull network firewalls can also be manipulated into being in a different state than the end host through trickery such as carefully chosen TTLs and fragment reassembly. Zhongjie Wang, Yue Cao, Zhiyun Qian, and Srikanth V. Krishnamurthy. 2017. Your state is not mine: A closer look at evading stateful Internet censorship. In *Internet Measurement Conference (IMC)*. ACM, (Nov. 2017) . Guestimating connections for connectionless protocols like UDP and ICMP can be fiddly and fragile.

Application proxies do more intrusive application specific filtering and can block common attacks on web applications by blocking requests that look like they are exploiting SQL injection etc.

Network Address Translation (NAT), designed to cope with insufficient IPv4 addresses blocks all incoming connections which has a firewalling effect. With IPv6 where every device can have a public IP again, home routers are supposed to provide a firewall that drops all incoming connections instead.

UPnP: Automatic network firewall hole punching, key enabler for IoT botnets as devices automatically expose vulnerable services.

Firewalls can also be used to redirect or re-write traffic to send it to a different port. For example, redirect port 80 to port 8080 for an application that cannot open port 80.

# Intrusion detection/prevention systems (IDS/IPS)

- Use deep packet inspection to detect or prevent known kinds of attacks
- Can have false positives
- Only blocks known kinds of attacks or know bad source addresses: probabilistic defence
- Internal network scanning for known vulnerabilities/unpatched systems

Because firewalls tend to block things on non-standard ports, everything ends up using port 80/443 and so deep packet inspection is used to work out what is really going on.

False positives: e.g. journal websites with long URLs are not trying to trigger buffer overflows. If the false positive rate is too high people stop caring.

On high load just IPSs can just let things through. Sometimes used for beancounting purposes to reduce IT risk for the risk register (having one can count, even if turned off). Such systems can become a single point of failure as the complex system sits between you and the Internet doing complex processing of untrusted data. How well these systems work depends to a large extent on the quality of the threat intelligence feeds.

IDS/IPS systems can use reputation (of IP addresses), signatures (of known attacks), and anomaly detection (abnormal behaviour).

Internal scanning: Find vulnerabilities before the bad guys do. Solution based on Nessus used by the University. Results are widely ignored.

# Penetration testing

- Pay someone to try and break into your network
- An external company (££££s)
- An internal red team (cheaper + more effective for larger organisations)
- Cross department/organisation war gaming

# VLANs

- Break the network up into separate virtual networks
- Different IP subnets in each VLAN
- Can deploy firewall rules between VLANs
- Keep finance division desktops and public facing servers separate

Can use this for things like guest wireless networks, BYOD networks, untrusted IoT devices. Can be connected with external users via VPNs (discussed later). Separate network for BMCs and management of devices.

# Don't allow spoofing[25]

- TCP's 3-way handshake verifies the source IP address
- UDP and raw IP do not verify the source IP address
- Spoofed source addresses can be used for attacks
- Don't allow spoofed source addresses out of your network (BCP38/SAVE)
- Bogons: source addresses that are never valid and indicate spoofing

[25] P. Ferguson and D. Senie. 2000. Network ingress filtering: Defeating denial of service attacks which employ IP source address spoofing. RFC 2827 (Best Current Practice: BCP38). Internet Engineering Task Force, (May 2000).

Source IP address is verified during the TCP 3-way handshake:

$$\begin{aligned} S \rightarrow D: &\quad \mathtt{SYN}_x \\ D \rightarrow S: &\quad \mathtt{SYN}_y, \mathtt{ACK}_{x+1} \\ S \rightarrow D: &\quad \mathtt{ACK}_{y+1} \end{aligned}$$

Only the third message starts data delivery, therefore data communication will only proceed after the claimed originator has confirmed the reception of a TCP sequence number in an ACK message. From then on, TCP will ignore messages with sequence numbers outside the confirmation window. In the absence of an eavesdropper, the start sequence number can act like an authentication nonce.

Negligent not to implement BCP38

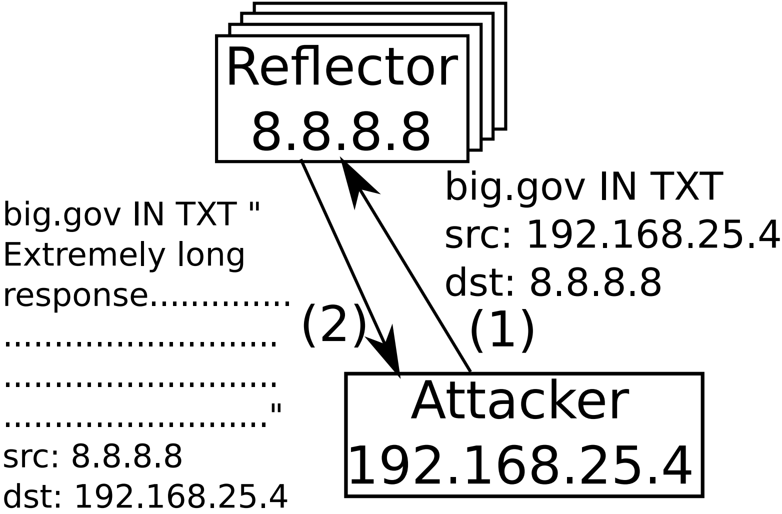
Bogons include private IP ranges such as 192.168.0.0/16, 10.0.0.0/8, or 172.16.0.0/12 and reserved ranges 224.0.0.0/4 (mulicast), 240.0.0.0/4 (future use).

# Distributed Denial of Service (DDoS)

- Overload networks or systems with spurious requests
- Blackmail, competitive advantage, revenge, and booters (DDoSaaS)

Anyone here been DDoSed? **[Audience participation]**

# UDP scanning finds reflectors



big.gov IN TXT
src: 192.168.25.4
dst: 8.8.8.8

big.gov IN TXT "
Extremely long
response.............
.........................
.........................
........................."
src: 8.8.8.8
dst: 192.168.25.4

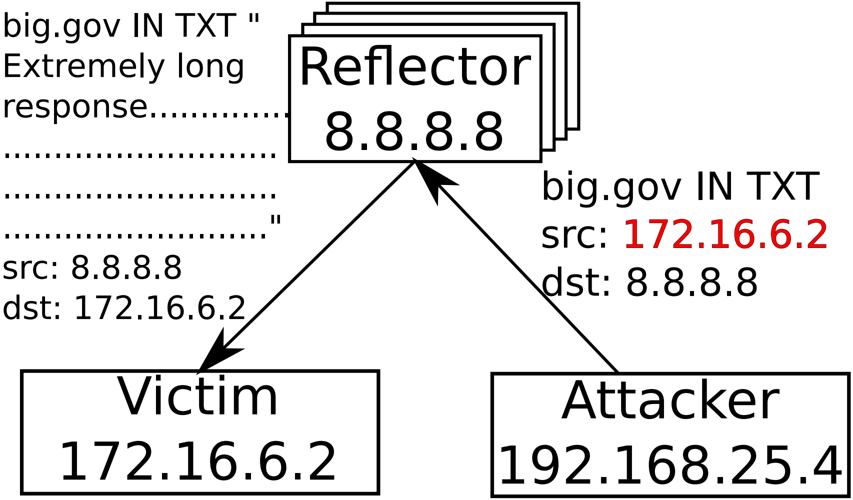
To conduct UDP amplification DDoS attacks the attacker first needs to find reflectors it can use to reflect off.

To do this it uses UDP in a standard way, sending out UDP packets and collecting the responses.

In this example it sends out a DNS packet, and when it finds a real reflector it gets a response back.

In this way by scanning the IPv4 space attackers can build up a list of all the reflectors they can use for attacks. This can be done in about 45 minutes on a fast connection and find thousands of millions of reflectors.

# UDP reflection attacks



big.gov IN TXT "
Extremely long
response.............
.........................
.........................
........................."
src: 8.8.8.8
dst: 172.16.6.2

Reflector
8.8.8.8

big.gov IN TXT
src: 172.16.6.2
dst: 8.8.8.8

Victim
172.16.6.2

Attacker
192.168.25.4

UDP reflection DDoS attacks exploit the fact that UDP (unlike TCP) does not verify the source IP address with a 3 way handshake. Hence, if an attacker can spoof the source IP address on the packets they send then the response will go to their victim.

In this example the attacker sends a DNS query to a resolver but spoofs the source IP address as the victim IP address. The much larger response goes to the victim.

The attacker can repeat this many times and over thousands of resolvers. This results in a large volume of traffic to the victim. The victim does not know the address of the attacker.

Most of the attacks using this method are from booters: DDoS as a service.

# TCP SYN floods

- Send lots of TCP SYN packets with spoofed source IPs
- Some implementations have finite number of half open TCP connections
- End points can mitigate with SYN-cookies but doesn't work for state-full firewalls in the middle.

# Botnet based attacks

- Build a network of bots and use them to directly send traffic
- Internet wide scanning and password or remote exploit based (e.g. Mirai)
- Malware infection of end-user-devices (drive by downloads, email based etc.)

# Web based amplification

- Wordpress XMLRPC callbacks can cause Wordpress instances to fetch the victim webpage
- Application layer reflection attacks in general

# Javascript in the browser

- Get browsers of visitors to third party sites to do the attack
- China vs. GitHub[26]

[26] Dan Goodin. 2015. Massive denial-of-service attack on GitHub tied to Chinese government. (Mar. 31, 2015). Retrieved Jan. 4, 2018 from https://arstechnica.com/information-technology/2015/03/massive-denial-of-service-attack-on-github-tied-to-chinese-government/.

# What just happened

Knowing that you are being attacked and what with helps you to respond. Good operators detect and respond in $<$30s

# Mitigation

- Bigger pipes? (1Tbit/s?)
- Traffic scrubbing
    - Boxes
    - BGP routing to provider
- Outsource websites etc. to providers that have mitigations in place
- Work out who is doing it and send them to the headteacher

All about quickly dropping the DDoS packets but not the normal user packets. Easy for floods on weird UDP ports, harder for TCP or application layer attacks.

Buy a box that sits on your network connection and is really good at dropping packets. However this is no good if your external connection is overloaded.

BGP: During an attack or always re-route traffic through a provider that scrubs traffic before passing it on.

Headteacher because many of those doing this are children.

# Honeypots

- Systems that pretend to be exploitable
- Monitor and waste attacker time
- Short scripts to full interaction VMs
- Mostly catch robots but also some humans
- How can abuse of the honeypot be prevented?

# Security protocols and trust models

Internet protocols that exist to provide security properties and the trust models they use.

# Transport Layer Security (TLS)

Successor to insecure Secure Sockets Layer (SSL). Provides security on top of TCP. Backwards compatible nightmare. The ‘s’ in ‘https’ or ‘imaps’

- Confidentiality (encrypted)
- Integrity (message authentication code)
- Authentication (public key cryptography)
- Hard to get right so test it: `https://www.ssllabs.com/ssltest/`

Who has configured TLS for a server? **[Audience participation]**

# TLS weaknesses[28]

- Everything that can go wrong has gone wrong
- Long list of attacks on all layers
- Writing correct implementations is **hard**[27]

[27] Benjamin Beurdouche, Karthikeyan Bhargavan, Antoine Delignat-Lavaud, Cédric Fournet, Markulf Kohlweiss, Alfredo Pironti, Pierre-Yves Strub, and Jean Karim Zinzindohoue. 2015. A Messy State of the Union: Taming the Composite State Machines of TLS. . In *IEEE Symposium on Security & Privacy*. http://research.microsoft.com/en-US/about/papers/taming-the-composite-state-machine-of-tls.pdf.

[28] Jeremy Clark and Paul C. van Oorschot. 2013. SoK: SSL and HTTPS: Revisiting past challenges and evaluating certificate trust model enhancements. *IEEE Symposium on Security and Privacy*, 511–525. IEEE Computer Society. http://www.ieee-security.org/TC/SP2013/papers/4977a511.pdf.

# Perfect Forward Secrecy (PFS)

- Negotiate new keys (using Diffe-Hellman) authenticated with old keys, destroy old keys.
- Later compromise does not allow decryption of old intercepted traffic.
- Prior compromise does not allow decryption of newer intercepted traffic.
- Key change in TLS configuration for a post-Snowden world.

ECDHE and DHE cipher modes (DHE is slower)

# Certificate authorities

- Certificate Authorities (CAs) are trusted to certify the identities associated with public keys.
- Public keys and identities are embedded in *certificates*
- Certificate authorities sign certificates

Public key encryption and signature algorithms allow the establishment of confidential and authenticated communication links with the owners of public/private key pairs.

Public keys still need to be reliably associated with identities of owners. In the absence of a personal exchange of public keys, this can be mediated via a trusted third party. Such a *certification authority* $C$ issues a digitally signed *public key certificate* in which $C$ confirms that the public key $K_A$ belongs to $A$ starting at time $T$ and that this confirmation is valid for the time interval $L$, and all this is digitally signed with $C$'s private signing key $K_C^{-1}$.

$$\mathrm{Cert}_C(A) = \{A, K_A, T, L\}_{K_C^{-1}}$$

Anyone who knows $C$'s public key $K_C$ from a trustworthy source can use it to verify the certificate $\mathrm{Cert}_C(A)$ and obtain a trustworthy copy of $A$'s key $K_A$ this way.

# Verifying certificates

Certificates must be verified

- Is the signature valid?
- Has the certificate expired?
- Has the certificate been revoked?

We can use the operator $\bullet$ to describe the extraction of $A$'s public key $K_A$ from a certificate $\mathrm{Cert}_C(A)$ with the certification authority public key $K_C$:

$$K_C \bullet \mathrm{Cert}_C(A) = \begin{cases} K_A & \text{if certificate valid} \\ \text{failure} & \text{otherwise} \end{cases}$$

The $\bullet$ operation involves not only the verification of the certificate signature, but also the validity time and other restrictions specified in the signature. For instance, a certificate issued by $C$ might contain a reference to an online *certificate revocation list* published by $C$, which lists all public keys that might have become compromised (e.g., the smartcard containing $K_A^{-1}$ was stolen or the server storing $K_A^{-1}$ was broken into) and whose certificates have not yet expired.
Certificate verification is fiddly hard to get right and many bugs have been found in implementations.
In general certificate revocation does not work and is not used. Certificate authorities cannot be trusted to keep their certificate revocation list (CRL) servers working and so if the server cannot be contacted it is assumed the certificate is not revoked. During an attack which intercepts TLS they can just block the access to the CRL server as well and use a revoked certificate.

# Certificate chains

- Certificates can be chained together
- CA root cert (kept offline) certifies CA intermediate cert
- CA intermediate cert certifies leaf cert
- leaf cert sent to clients with a copy of the intermediate cert
- Client verifies intermediate cert and then uses that to verify leaf cert

Public keys can also be verified via several trusted intermediaries in a *certificate chain*:

$$K_{C_1} \bullet \mathrm{Cert}_{C_1}(C_2) \bullet \mathrm{Cert}_{C_2}(C_3) \bullet \cdots \bullet \mathrm{Cert}_{C_{n-1}}(C_n) \bullet \mathrm{Cert}_{C_n}(B) = K_B$$

$A$ has received directly a trustworthy copy of $K_{C_1}$ (which many implementations store locally as a certificate $\mathrm{Cert}_A(C_1)$ to minimise the number of keys that must be kept in tamper-resistant storage).

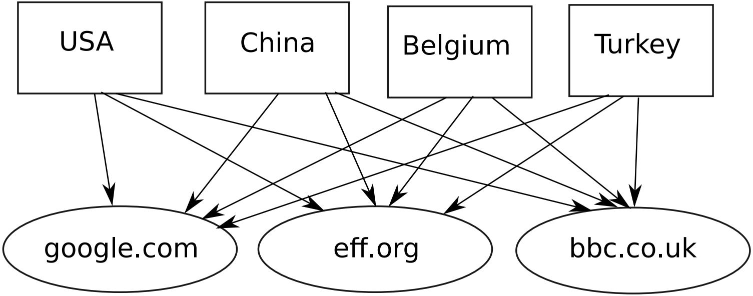
Certification authorities can be made part of a hierarchical tree, in which members of layer $n$ verify the identity of members in layer $n-1$ and $n+1$. For example layer 1 can be a national CA, layer 2 the computing services of universities and layer 3 the system administrators of individual departments.

Practical example: $A$ personally receives $K_{C_1}$ from her local system administrator $C_1$, who confirmed the identity of the university's computing service $C_2$ in $\mathrm{Cert}_{C_1}(C_2)$, who confirmed the national network operator $C_3$, who confirmed the IT department of $B$'s employer $C_3$ who finally confirms the identity of $B$. An online directory service allows $A$ to retrieve all these certificates (plus related certificate revocation lists) efficiently.

Putting all the certificates on the servers in the right place and in the right order is fiddly and people often get it wrong.

# CAs: Trust hundreds of companies all over the world completely?

- About 600 organisations have trusted signing keys which can be used to sign a certificate for use with TLS/SSL for any domain.
- Including the governments of various countries which don't get on.
- Widely considered broken, but no real alternative.



Certificate Authority (CA)

Let's Encrypt `https://letsencrypt.org/` is a CA that provides free certificates automatically after verification of control of the domain using the ACME protocol. Supported packages for all good operating systems.

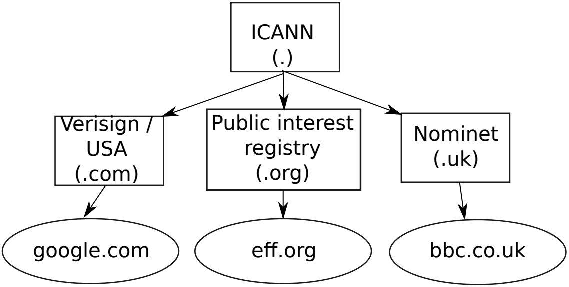
# Certificate transparency

- Idea: Audit the CAs so that bad certificates can be detected
- Append only verifiable log (Merkle Tree Hashes)
- Ship evidence of inclusion in the log with the certificate
- Reject certificates not included

`https://www.certificate-transparency.org/what-is-ct`

# DNSSEC: Trust the root and the DNS hierarchy?

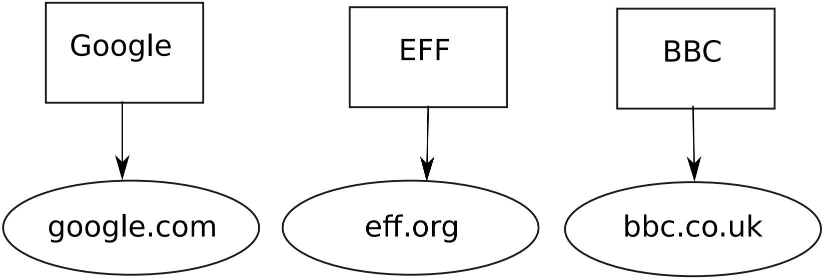
- DNS root signs keys for .uk, .com, .org etc.
- .uk signs .ac.uk, .ac.uk signs cam.ac.uk etc.
- ‘only’ trust path to the root but are DNS registrars going to do this job well? Or even better than the CAs?



DANE proposes the use of DNSSEC to replace CAs but no browsers support this. Viktor Dukhovni and Wes Hardaker. 2015. RFC 7671: The DNS-Based Authentication of Named Entities (DANE) Protocol: Updates and Operational Guidance. Tech. rep. IETF, (Oct. 2015), 1–33

# Trust the certificate the app has hard coded

- Each app contains hard coded certificates for all the public-keys it needs to rely on.
- Any other key rejected.
- Keys changed by updating the app.



# SSH: Secure shell

- Secure remote shell
- Public key authentication (password bruteforcing -> fail2ban)
- Port forwarding
- X forwarding

Hands up if you have used SSH. **[Audience participation]**

This is how most servers are controlled and configured. Widely supported and easy to setup and use.

# TOFU: Trust on first use: Trust the first key you ever saw for that address? I

- The first time you see a key for an address you trust it.
- If it ever changes then scream.
- But how do we legitimately change keys?

The default trust model for SSH is TOFU.

The `No matching host key fingerprint found in DNS.` is because it does an SSHFP record DNS lookup which if DNSSEC signed would allow the use of the DNSSEC trust model for first use.

I hit one of these false positives several times a year but have yet to hit a true positive.

# TOFU: Trust on first use: Trust the first key you ever saw for that address? II

Bleats the first time (everyone just types yes).

```
$ ssh sshtest.dtg.cl.cam.ac.uk
The authenticity of host 'sshtest.dtg.cl.cam.ac.uk
(128.232.20.65)' can't be established.
ECDSA key fingerprint is 64:49:35:b8:8d:7c:ae:e4:c9:e0:3a:c5:8
No matching host key fingerprint found in DNS.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'sshtest.dtg.cl.cam.ac.uk,
128.232.20.65' (ECDSA) to the list of known hosts.

drt24@sshtest:~$ logout
Connection to sshtest.dtg.cl.cam.ac.uk closed.
```

# TOFU: Trust on first use: Trust the first key you ever saw for that address? III

Silent on subsequent connections (unless the key has changed)

```
$ ssh sshtest.dtg.cl.cam.ac.uk
drt24@sshtest:~$ logout
Connection to sshtest.dtg.cl.cam.ac.uk closed.
```

# TOFU: Trust on first use: Trust the first key you ever saw for that address? IV

```
$ ssh sshtest.dtg.cl.cam.ac.uk
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@      WARNING: POSSIBLE DNS SPOOFING DETECTED!         @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
The RSA host key for sshtest.dtg.cl.cam.ac.uk has changed,
and the key for the corresponding IP address 128.232.21.44
is unknown. This could either mean that
DNS SPOOFING is happening or the IP address for the host
and its host key have changed at the same time.
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@   WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!   @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
```

# TOFU: Trust on first use: Trust the first key you ever saw for that address? V

```
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-mid
It is also possible that a host key has just been changed.
The fingerprint for the RSA key sent by the remote host is
SHA256:lwyk+7Ro0tl5P3RCdGUBUpdYNr/aANipe5CWu6N/Q14.
Please contact your system administrator.
Add correct host key in /home/drt24/.ssh/known_hosts to get rid
Offending ECDSA key in /home/drt24/.ssh/known_hosts:255
  remove with:
 ssh-keygen -f "/home/drt24/.ssh/known_hosts" -R sshtest.dtg.
RSA host key for sshtest.dtg.cl.cam.ac.uk has changed and you ha
Host key verification failed.
```

# TOFU: Trust on first use: Trust the first key you ever saw for that address? VI

Almost always a false positive so:

```
$ ssh-keygen -f "/home/drt24/.ssh/known_hosts" -R sshtest.dtg.
# Host sshtest.dtg.cl.cam.ac.uk found: line 255
/home/drt24/.ssh/known_hosts updated.
Original contents retained as /home/drt24/.ssh/known_hosts.old
$ ssh sshtest.dtg.cl.cam.ac.uk
The authenticity of host 'sshtest.dtg.cl.cam.ac.uk (128.232.21
RSA key fingerprint is SHA256:lwyk+7Ro0tl5P3RCdGUBUpdYNr/aANip
No matching host key fingerprint found in DNS.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'sshtest.dtg.cl.cam.ac.uk,128.232.
drt24@sshtest:~$
```

# VPNs: Virtual Private Networks

- Securely connect two networks over an untrusted intermediate network
- Encrypt entire packets including network layer
- Difficult to configure correctly
- Protocols such as IPSec/L2TP
- Implementations such as strongswan

Virtual Private Networks (VPNs) can be used to securely connect networks or individual machines over an untrusted network. They encrypt entire packets including the network layer (IP/TCP headers), and so hide both the content and destination of traffic. These encrypted packets then have new IP headers with the destination being the VPN endpoint.

There are a large range of protocols and some of them are old and insecure. Some protocols use TLS's authentication protocol. Two-way authentication is difficult without client certificates, consider a VPN client sending a plaintext password to the VPN server to authenticate, how does it know it is not a MITM. Server certificates might not contain the right name (poor configuration), might need a strange root certificate, revocation checks might not work until after connection.

Clients might not check certificates.

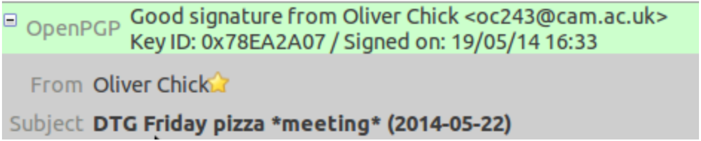
Thanks to Malcolm Scott for input on VPNs.

# Secure messaging

- People want to communicate
- How do we do so securely?

# Sign and encrypt email with GPG

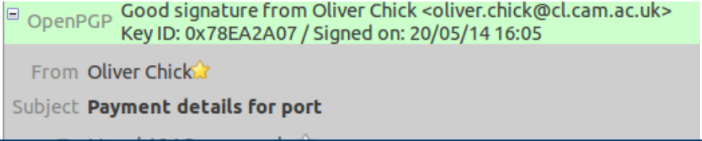
- Sign all your email using GPG: Enigmail for Thunderbird, Evolution has built in support and various options on other platforms.
- Key can be verified based on prior communications.
- Can encrypt emails when you share keys



⊟ OpenPGP Good signature from Oliver Chick <oc243@cam.ac.uk>
Key ID: 0x78EA2A07 / Signed on: 19/05/14 16:33

From Oliver Chick

Subject **DTG Friday pizza *meeting* (2014-05-22)**



⊟ OpenPGP Good signature from Oliver Chick <oliver.chick@cl.cam.ac.uk>
Key ID: 0x78EA2A07 / Signed on: 20/05/14 16:05

From Oliver Chick

Subject **Payment details for port**

I have been doing this for years without any real issues. It works well for me, it might for you but it doesn't work for the general population.

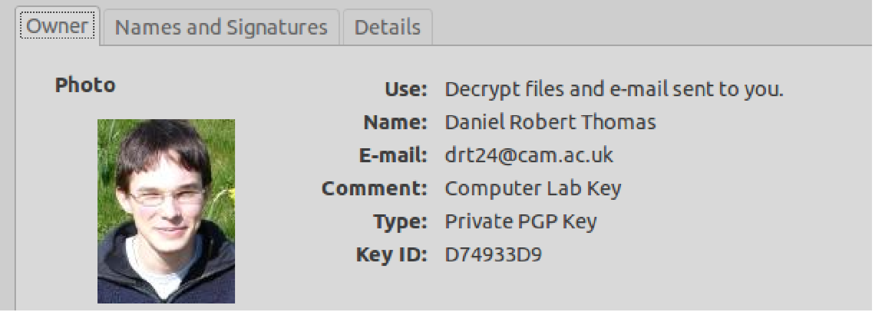
I can authenticate Oliver's request to pay money into his bank account against his use of the same key he used to send me ordinary emails in the past.

# Use GPG to manage your personal public-keys

```
 $ gpg --fingerprint --list-sigs D74933D9
pub   4096R/D74933D9 2012-04-19 [expires: 2017-10-31]
      Key fingerprint = 5017 A1EC 0B29 08E3 CF64  7CCD 5514 35D5 D749 33D9
uid                  Daniel Robert Thomas (Computer Lab Key) <drt24@cam.ac.uk>
sig 3        D74933D9 2012-04-19  Daniel Robert Thomas (Computer Lab Key) <drt24@cam.ac.uk>
sig          78EA2A07 2012-11-12  Oliver Chick <oliver.chick@cl.cam.ac.uk>
sig          18EB83B1 2012-04-26  Daniel Robert Thomas (Cambridge University Email) <drt24@cam.ac.uk>
sig 2        5E2A64A6 2013-08-16  Steven Murdoch <steven@murdomedia.net>
[snip]
uid                  [jpeg image of size 3954]
sig 3        D74933D9 2012-04-19  Daniel Robert Thomas (Computer Lab Key) <drt24@cam.ac.uk>
[snip]
uid                  Daniel Robert Thomas <drt24@danielkirsty.me.uk>
sig 3        D74933D9 2013-11-20  Daniel Robert Thomas (Computer Lab Key) <drt24@cam.ac.uk>
sub   4096R/60016489 2012-04-19 [expires: 2017-10-31]
sig          D74933D9 2012-04-19  Daniel Robert Thomas (Computer Lab Key) <drt24@cam.ac.uk>
sub   2048R/52C058CE 2012-06-27
sig          D74933D9 2012-06-27  Daniel Robert Thomas (Computer Lab Key) <drt24@cam.ac.uk>
```

GPG is the default program which people use to manage their personal key-pairs, it also has various GUIs such as `seahorse` shown here.
You can use it to generate keys, and then use them for encrypting, decrypting, signing and verifying.
This output shows various things.
Various data items are associated with the public key and bound by signatures. It has a creation and expiry date, a fingerprint which can be used to uniquely identify it. It has uids which are the identities which the key asserts are associated with it, it has an image and some subkeys.
It has been signed by various other people's keys who then assert that the uids which they signed are correct.

# GPG GUIs



Photo

**Use:** Decrypt files and e-mail sent to you.

**Name:** Daniel Robert Thomas

**E-mail:** drt24@cam.ac.uk

**Comment:** Computer Lab Key

**Type:** Private PGP Key

**Key ID:** D74933D9

My GPG keys include as an identifier a photo of my 16 year old self. This is becoming a progressively less useful identifier.

# Group discussion: When should you sign someone else's key?

**[Audience participation]**
2.5 minutes discussion in small groups then feed back.

- How do you prove someone is identified by the uids they claim?
- What tricks could you use break those mechanisms?
- How could you defend against those tricks?
- Who is your adversary?

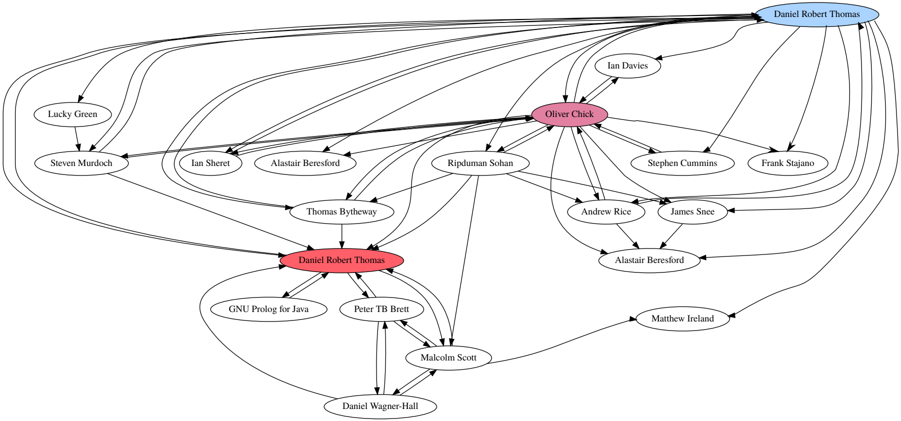
Do they go by that name? Does email sent to that address, encrypted arrive? Verification of identity documents?

Government issued ID, fake ID, going under false name, DNS attacks, email interception.

Only verify for people you know personally over an extended period, secure networks

Joe Blogs or Prof. Moriarty?

# You trust people you trust? And the people they trust? Web of Trust



This is the model commonly used with GPG where you use the interconnected web of signatures to verify keys you have not seen before.

Unfortunately trust is not transitive and so very dense networks are required.

The network shown is the interesting part of my WoT, despite many years work it is still rather small and not particularly well connected.

This does not scale to the whole world.

This graph shows two of my keys, my work one and home one. It also shows Oliver Chick's key we sign keys for people in our research group.

I can trust signatures made on keys by my home key as it is trusted by my work key. Hence I can trust Malcolm Scott's key even though I don't have a direct link.

# GPG best practices

- Use a 3072bit or better RSA key using sha512 hashing
- Schedule automatic refresh of keyring from keyservers (so you get revocations) (use parcimonie if you want to keep the contents of your keyring private)
- Verify fingerprints
- Don't use keyid (32bits long and easily brute forced)
- Set an expiration date (in case you lose the key) and a reminder to extend it (so it doesn't actually expire)
- Generate a revocation certificate

`https://riseup.net/en/security/message-security/openpgp/best-practices`

# OTR for secure chat

- Perfectly forward secret chat communication.
- Implementations for various IM platforms
- WhatsApp is the most popular
- Signal for Android and iOS makes key changes explicit
- iMessage does something similar

`https://otr.cypherpunks.ca/`

`https://signal.org/`

Off-the-Record (OTR) Messaging allows you to have private conversations over instant messaging by providing:

Encryption No one else can read your instant messages.

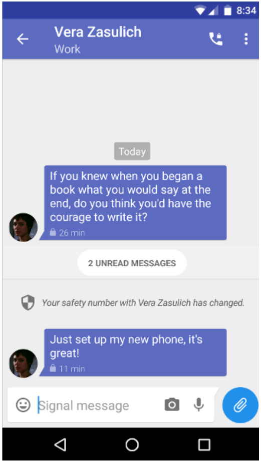
Authentication You are assured the correspondent is who you think it is.

Deniability The messages you send do not have digital signatures that are checkable by a third party. Anyone can forge messages after a conversation to make them look like they came from you. However, during a conversation, your correspondent is assured the messages he sees are authentic and unmodified.

Perfect forward secrecy If you lose control of your private keys, no previous conversation is compromised.

If a messaging provider can't intercept the messages that your are sending then they don't need to deal with the horrendous legal complexities surrounding warrants from every jurisdiction as the answer is always "We don't have that information".

# Identity key changes



Vera Zasulich
Work

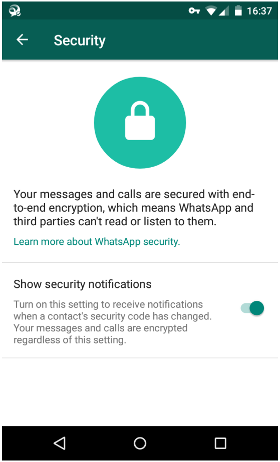
Today

If you knew when you began a book what you would say at the end, do you think you'd have the courage to write it?
26 min

2 UNREAD MESSAGES

*Your safety number with Vera Zasulich has changed.*

Just set up my new phone, it's great!
11 min



Security

Your messages and calls are secured with end-to-end encryption, which means WhatsApp and third parties can't read or listen to them.

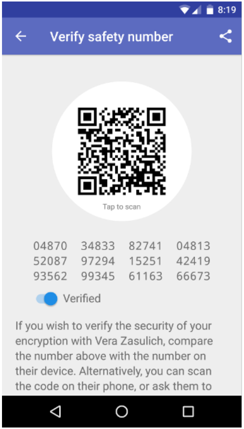
Learn more about WhatsApp security.

Show security notifications

Turn on this setting to receive notifications when a contact's security code has changed. Your messages and calls are encrypted regardless of this setting.

Signal image from:
`https://signal.org/blog/verified-safety-number-updates/`
Signal displays safety number change messages by default while WhatsApp doesn't but has a setting for turning it on.

# Safety number verification



8:19

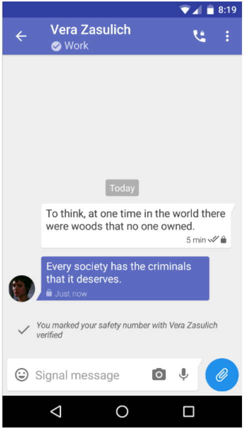
Verify safety number

Tap to scan

```
04870   34833   82741   04813
52087   97294   15251   42419
93562   99345   61163   66673
```

Verified

If you wish to verify the security of your encryption with Vera Zasulich, compare the number above with the number on their device. Alternatively, you can scan the code on their phone, or ask them to



8:19

Vera Zasulich

Work

Today

To think, at one time in the world there were woods that no one owned.

5 min

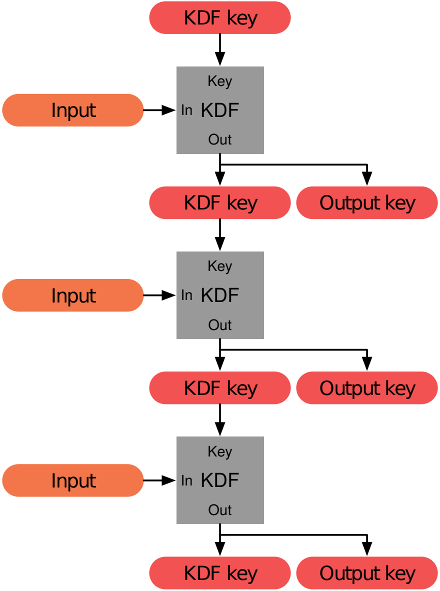
Every society has the criminals that it deserves.

Just now

*You marked your safety number with Vera Zasulich verified*

Signal message

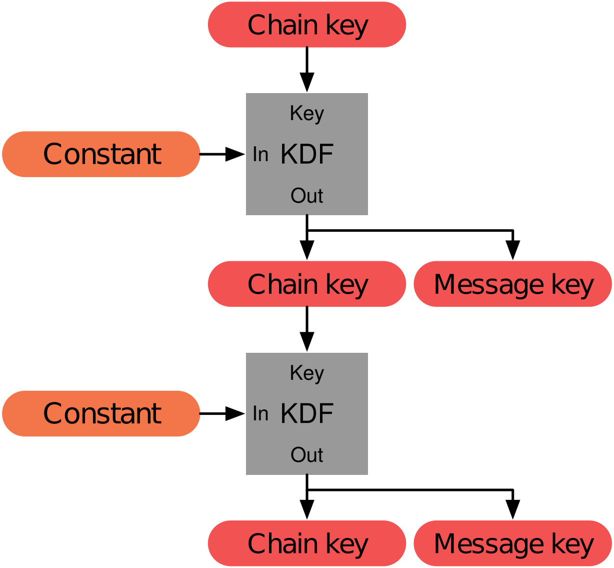
Signal has a way of verifying safety numbers while WhatsApp does not.


KDF key
Key
Input
In
KDF
Out
KDF key
Output key
Key
Input
In
KDF
Out
KDF key
Output key
Key
Input
In
KDF
Out
KDF key
Output key

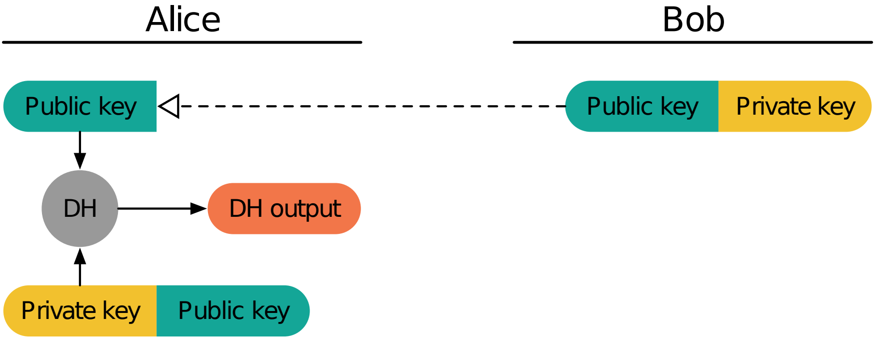
We will flick through the double ratchet process used in Signal to give you an idea of how Perfect Forward Secrecy is achieved. The same protocol is used for WhatsApp. `https://signal.org/docs/specifications/doubleratchet/doubleratchet.pdf` Diagrams are all from this document.

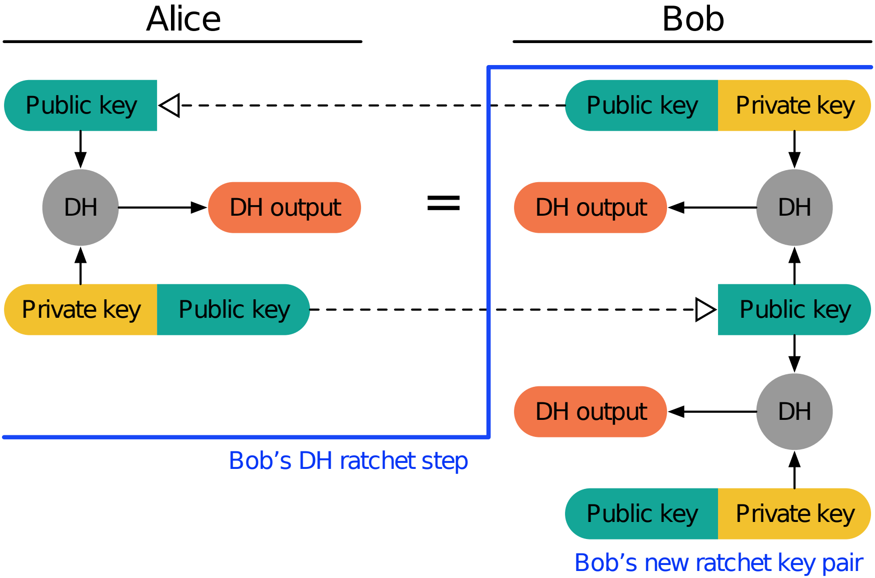
A KDF chain provides (quoting the above reference):

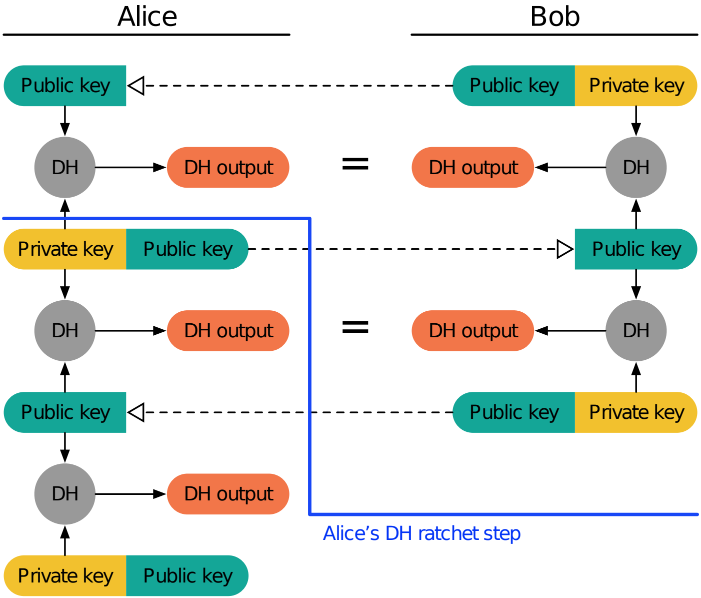
- Resilience: The output keys appear random to an adversary without knowledge of the KDF keys. This is true even if the adversary can control the KDF inputs.
- Forward security: Output keys from the past appear random to an adversary who learns the KDF key at some point in time.
- Break-in recovery: Future output keys appear random to an adversary who learns the KDF key at some point in time, provided that future inputs have added sufficient entropy.

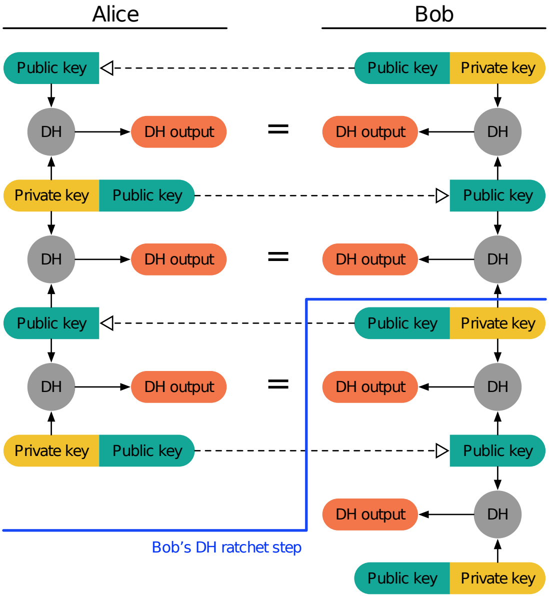

Chain key
Key
Constant
In
KDF
Out
Chain key
Message key
Key
Constant
In
KDF
Out
Chain key
Message key

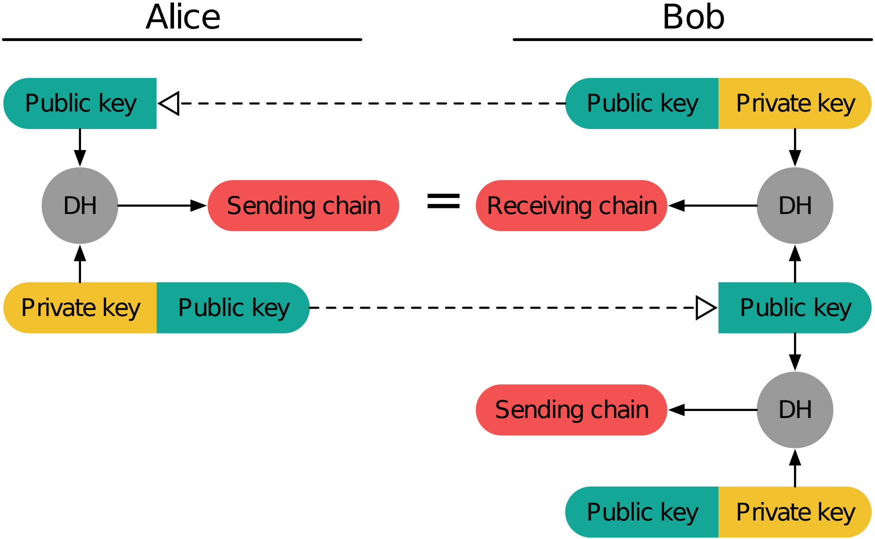
# Diffie-Hellman ratchet

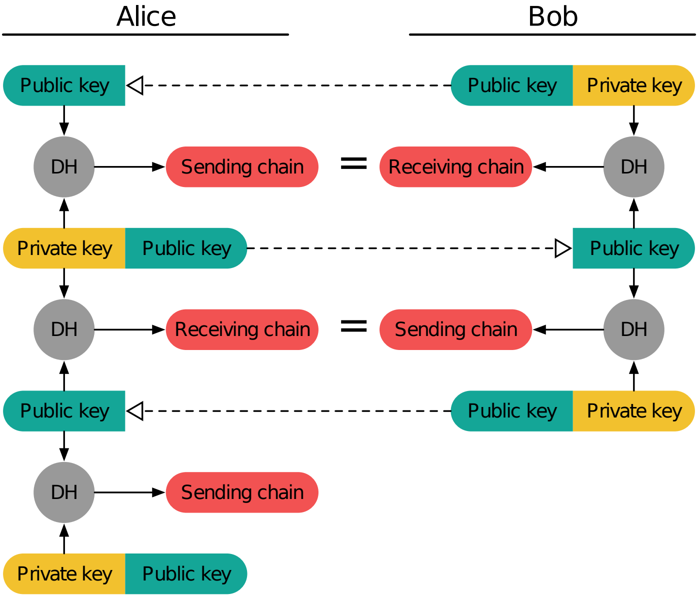


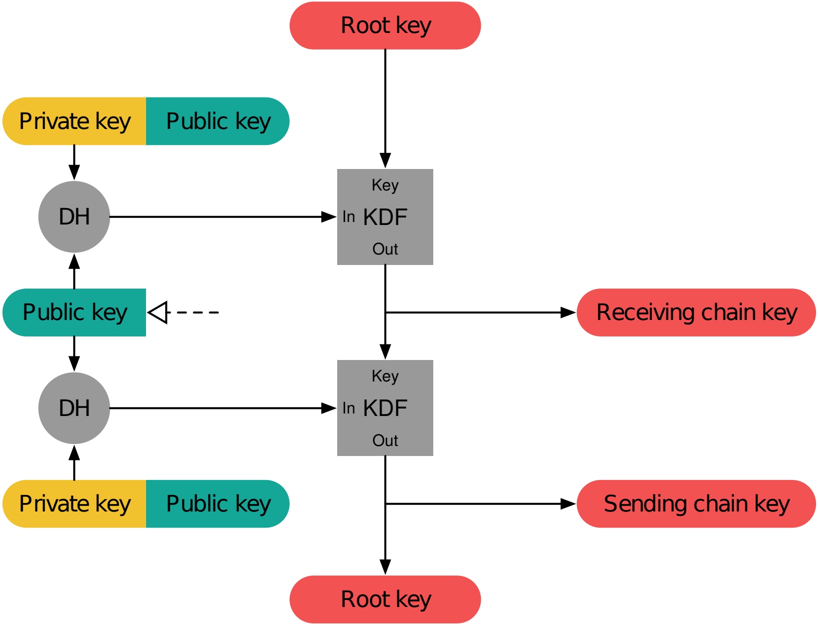

Alice
Bob
Public key
Public key
Private key
DH
DH output
=
DH output
DH
Private key
Public key
Public key
DH output
DH
Bob’s DH ratchet step
Public key
Private key
Bob’s new ratchet key pair

Alice
Bob
Public key
Public key
Private key
DH
DH output
=
DH output
DH
Private key
Public key
Public key
DH
DH output
=
DH output
DH
Public key
Public key
Private key
DH
DH output
Alice’s DH ratchet step
Private key
Public key

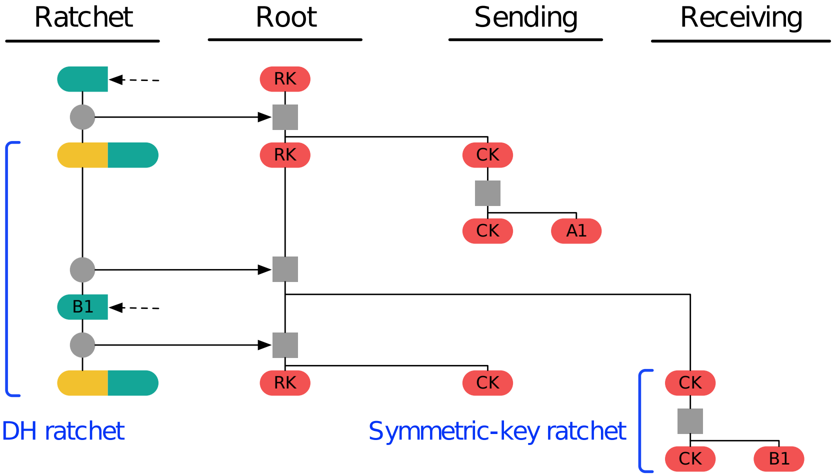

Alice
Bob
Public key
Public key
Private key
DH
DH output
=
DH output
DH
Private key
Public key
Public key
DH
DH output
=
DH output
DH
Public key
Public key
Private key
DH
DH output
=
DH output
DH
Private key
Public key
Public key
DH output
DH
Public key
Private key
Bob’s DH ratchet step


Alice
Bob
Public key
Public key
Private key
DH
Sending chain
=
Receiving chain
DH
Private key
Public key
Public key
Sending chain
DH
Public key
Private key


Alice
Bob
Public key
Public key
Private key
DH
Sending chain
=
Receiving chain
DH
Private key
Public key
Public key
DH
Receiving chain
=
Sending chain
DH
Public key
Public key
Private key
DH
Sending chain
Private key
Public key


Root key
Private key
Public key
DH
Key
In
KDF
Out
Public key
Receiving chain key
DH
Key
In
KDF
Out
Private key
Public key
Sending chain key
Root key

# Double Ratchet



# Thank you!

Daniel.Thomas@cl.cam.ac.uk

`https://www.cl.cam.ac.uk/~drt24/`

`GPG: 5017 A1EC 0B29 08E3 CF64 7CCD 5514 35D5 D749 33D9`

We offer PhDs in solving or finding interesting security problems and there are two masters courses in security in Part III (R209 and R210).

References:

[1] Anne Adams and Martina Angela Sasse. 1999. Users are not the enemy. *Communications of the ACM*, 42, 12, 40–46. ACM.

[2] James P. Anderson. 1972. Computer security technology planning study. Tech. rep. Electronic Systems Division, Air Force Systems Command, Hanscom Field, Bedford, MA, (Oct. 1972). http://seclab.cs.ucdavis.edu/projects/history/CD-1/ande72.pdf.

[3] Ross Anderson. 2017. Banks biased against black fraud victims. (Jan. 2017). Retrieved Feb. 8, 2018 from https://www.lightbluetouchpaper.org/2017/01/12/banks-biased-against-black-fraud-victims/.

[4] Ross Anderson. 2008. *Security Engineering: A guide to building dependable distributed systems*. (2nd ed.). Wiley. ISBN: 978-0-470-06852-6. https://www.cl.cam.ac.uk/~rja14/book.html.

[5] Ross Anderson, Chris Barton, Rainer Böhme, Richard Clayton, Michel J. G. van Eeten, Michael Levi, Tyler Moore, and Stefan Savage. 2012. Measuring the cost of cybercrime. In *Workshop on the Economics of Information Security*. Springer, Berlin, Germany, 265–300. ISBN: 978-3-642-39497-3.

[6] Adam Beautement, M. Angela Sasse, and Mike Wonham. 2008. The compliance budget: Managing security behaviour in organisations. *Proceedings of the New Security Paradigms Workshop (NSPW)*, 47–58. ACM.

[7] D.E. Bell and L.J. LaPadula. 1973. Secure computer systems: Mathematical Foundations. Tech. rep. Electronic Systems Division, Air Force Systems Command, United States Air Force, L.G. Hanscom Field, Bedford, MA, (Nov. 1973). `http://www.dtic.mil/docs/citations/AD0770768`.

[8] Benjamin Beurdouche, Karthikeyan Bhargavan, Antoine Delignat-Lavaud, Cédric Fournet, Markulf Kohlweiss, Alfredo Pironti, Pierre-Yves Strub, and Jean Karim Zinzindohoue. 2015. A Messy State of the Union: Taming the Composite State Machines of TLS. In *IEEE Symposium on Security & Privacy*. http://research.microsoft.com/en-US/about/papers/taming-the-composite-state-machine-of-tls.pdf.

[9] Joseph Bonneau. 2012. Authentication is machine learning. (Dec. 2012). Retrieved June 26, 2013 from http://www.lightbluetouchpaper.org/2012/12/14/authentication-is-machine-learning/.

[10] Joseph Bonneau. 2011. Getting Web Authentication Right A Best-Case Protocol for the Remaining Life of Passwords. *Security Protocols XIX*, 7114, 98–104. Springer.

[11] Joseph Bonneau, Cormac Herley, Paul C. van Oorschot, and Frank Stajano. 2012. The quest to replace passwords: A framework for comparative evaluation of web authentication schemes. In *IEEE Symposium on Security and Privacy*.

[12] Joseph Bonneau and Stuart Schechter. 2014. Towards Reliable Storage of 56-bit Secrets in Human Memory. *USENIX Security Symposium (USENIX Security)*.

[13] Robert B. Cialdini. 2014. *Influence: science and practice*. (5th ed.). Pearson Education Limited, Harlow, Essex. ISBN: 9781292035499.

[14] Jeremy Clark and Paul C. van Oorschot. 2013. SoK: SSL and HTTPS: Revisiting past challenges and evaluating certificate trust model enhancements. *IEEE Symposium on Security and Privacy*, 511–525. IEEE Computer Society. `http://www.ieee-security.org/TC/SP2013/papers/4977a511.pdf`.

[15] Joe Deblasio, Stefan Savage, Geoffrey M. Voelker, and Alex C. Snoeren. 2017. Tripwire: Inferring Internet Site Compromise. In *Internet Measurement Conference (IMC)*. ACM, London, UK, (Nov. 2017).

[16] Viktor Dukhovni and Wes Hardaker. 2015. RFC 7671: The DNS-Based Authentication of Named Entities (DANE) Protocol: Updates and Operational Guidance. Tech. rep. IETF, (Oct. 2015), 1–33.

[17] P. Ferguson and D. Senie. 2000. Network ingress filtering: Defeating denial of service attacks which employ IP source address spoofing. RFC 2827 (Best Current Practice: BCP38). Internet Engineering Task Force, (May 2000).

[18] Cormac Florencio, Dinei and Herley. 2010. Where do security policies come from? In *Symposium on Usable Privacy and Security (SOUPS)*. ACM, (July 2010).

[19] Dan Goodin. 2015. Massive denial-of-service attack on GitHub tied to Chinese government. (Mar. 31, 2015). Retrieved Jan. 4, 2018 from `https://arstechnica.com/information-technology/2015/03/massive-denial-of-service-attack-on-github-tied-to-chinese-government/`.

[20] Gerwin Klein et al. 2009. seL4: Formal verification of an OS kernel. In *Proceedings of the ACM SIGOPS 22nd Symposium on Operating System Principles*. ACM, Big Sky, Montana, USA, (Oct. 2009), 207–220. ISBN: 9781605587523.

[21] Martin Kleppmann. 2011. Accounting for Computer Scientists. (Mar. 2011). Retrieved Jan. 29, 2018 from http://martin.kleppmann.com/2011/03/07/accounting-for-computer-scientists.html, https://perma.cc/Z66K-YYJT.

[22] Kevin D. Mitnick and William L. Simon. 2002. *The art of deception: Controlling the human element of security*. Wiley. ISBN: 978076454280.

[23] Kevin D. Mitnick and William L. Simon. 2005. *The art of intrusion: The real stories behind the exploits of hackers, intruders & deceivers*. Wiley. ISBN: 9780764569593.

[24] Tyler Moore and Richard Clayton. 2007. Examining the impact of website take-down on phishing. In *APWG eCrime Researchers Summit*. ACM, 1–13. ISBN: 9781595939398.

[25] Robert Morris and Ken Thompson. 1979. Password security: A case history. *Communications of the ACM*, 22, 11, 594–597. ACM.

[26] Steven J. Murdoch and Angela Sasse. 2017. Should you phish your own employees? (Aug. 22, 2017). Retrieved Jan. 4, 2018 from https://www.benthamsgaze.org/2017/08/22/should-you-phish-your-own-employees/.

[27] Shishir Nagaraja and Ross Anderson. 2009. The snooping dragon: social-malware surveillance of the Tibetan movement. Tech. rep. 746. Computer Laboratory, (Mar. 2009), 1–12. https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-746.html.

[28] Cabinet Office. 2013. Government Security Classifications. Tech. rep. (Oct. 2013). https://www.gov.uk/government/publications/government-security-classifications.

[29] Andreas Pashalidis and Chris J. Mitchell. 2003. A taxonomy of single sign-on systems. In *Australasian Conference on Information Security and Privacy (ACISP)*. Vol. LNCS 2727. Springer, (July 2003), 249–264.

[30] Oriana Riva, Chuan Qin, Karin Strauss, and Dimitrios Lymberopoulos. 2012. Progressive authentication: Deciding when to authenticate on mobile phones. *USENIX Security Symposium*. USENIX.

[31] Frank Stajano and Paul Wilson. 2011. Understanding scam victims. *Communications of the ACM*, 54, 3, 70–75. ACM.

[32] Daniel R. Thomas and Alastair R. Beresford. 2014. Better authentication: Password revolution by evolution. In *Security Protocols XXII*. Vol. LNCS 8809. Springer, Cambridge, UK, (Mar. 2014), 130–145. ISBN: 978-3-319-12399-8.

[33] Zhongjie Wang, Yue Cao, Zhiyun Qian, and Srikanth V. Krishnamurthy. 2017. Your state is not mine: A closer look at evading stateful Internet censorship. In *Internet Measurement Conference (IMC)*. ACM, (Nov. 2017).

[34] Robert N M Watson et al. 2015. CHERI: A hybrid capability-system architecture for scalable software compartmentalization. *Symposium on Security and Privacy*, 20–37. IEEE. ISSN: 10816011.

[35] Alma Whitten and J.D. Tygar. 1999. Why Johnny Can't Encrypt. In *USENIX Security Symposium*. (Aug. 1999), 679–702. http://www.doug-tygar.com/papers/Why_Johnny_Cant_Encrypt/OReilly.pdf.

[36] Maurice V. Wilkes. 1968. *Time-sharing computer systems*. MacDonald & Co. ISBN: 0356024261.