Non-User Authentication

Budi Arief
b.arief@kent.ac.uk

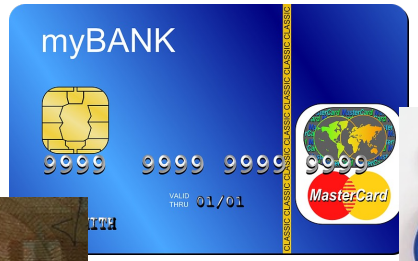
Based on slides by Shujun Li

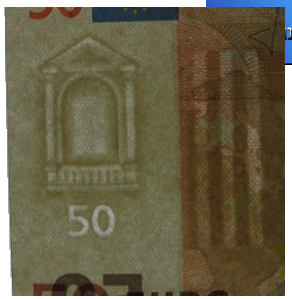
Outline

- Product Authentication
- Message Authentication (Integrity Verification)
- Server Authentication
- Software Identification (Fingerprinting)
- Authentication Protocols in Distributed Systems (Beyond User Authentication)
- Humanness Authentication (CAPTCHA)
- Summary

Physical products

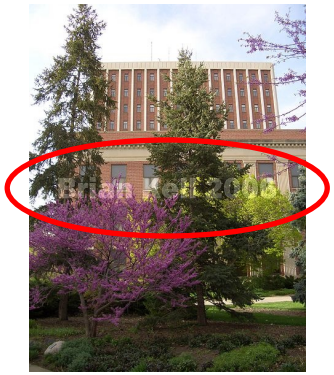
- Unique serial number (in the form of printed text, 1-D barcode, QR code)
  - Need to use an online or telephone service to find out if the serial number is valid and not reused/copied.
- Security hologram labels
- RFID chips
- Paper watermarks
- …
- These are not actually verifying the product itself.





Digital products: digital watermarking

- A digital watermark is a **digital artefact embedded into** a cover work (which is also a **digital artefact**).
  - The cover can be **anything**: texts, images, videos, audio channels/files, databases, 3D models, network traffic, software, …
  - The watermark can be **anything**: a textual message, an image, a random number (like an ID) represented as a number of bits, …
- The process of embedding a digital watermark is called digital watermarking.
- The watermark may be perceptible or imperceptible.



Outline

- Product Authentication
- Message Authentication (Integrity Verification)
- Server Authentication
- Software Identification (Fingerprinting)
- Authentication Protocols in Distributed Systems (Beyond User Authentication)
- Humanness Authentication (CAPTCHA)
- Summary

From my slides (2019-2020): MACs

Example: MAC

- A cryptographic *Message Authentication Code* (MAC) is an additional tag added to a message to authenticate it
- A MAC algorithm accepts as input a secret key and an arbitrary-length message to be authenticated, and outputs a MAC
- The MAC value protects both a message's *integrity* as well as its *authenticity*:
  - By allowing verifiers (who also possess the secret key) to detect any changes to the message content

From my slides (2019-2020): MACs

MAC (contd.)

- While MAC functions are similar to cryptographic hash functions, MACs differ from digital signatures
  - MAC values are both generated and verified using *the same* secret key
  - This implies that the sender and receiver of a message must agree on keys before initiating communications, as is the case with *symmetric encryption*
  - MAC algorithms can be constructed from crypto hash functions or from block cipher algorithms

From my slides (2019-2020): MACs

MAC (contd.)

- Suppose that Alice and Bob share the secret key *k*. Then Alice can authenticate a message *m* to Bob by computing:
  - *Hash(k,m)* and appending this to *m*, i.e.:

    Alice → Bob: *Hash(k,m),m*
- Now Bob can apply the keyed hash to the *m* term and check that it agrees with the tag term
- Without knowledge of *k*, it should be extremely unlikely that an adversary can find a message text, say *m′*, and compute a tag that will match

  Hash(k, m′)
  - Probability is essentially *1/#(range Hash)*, e.g. $(½)^{128}$ if the hash function gives 128 bit outputs

MAC and Digital Signature

- But how would Bob know if Alice is indeed the sender of the message? ➔ Digital Signatures

4. Protocols, Applications and Attacks

Digital Signatures

- These are the digital analogues of conventional signatures
- They are intended to confirm the *origin* and *authenticity* of a message

From my slides (2019-2020):
Digital Signatures

Digital Signatures (contd.)

- Suppose that Alice has secret key SK_A and public key PK_A
- She can sign a message M as follows:
  - Alice encrypts M this with her secret key:

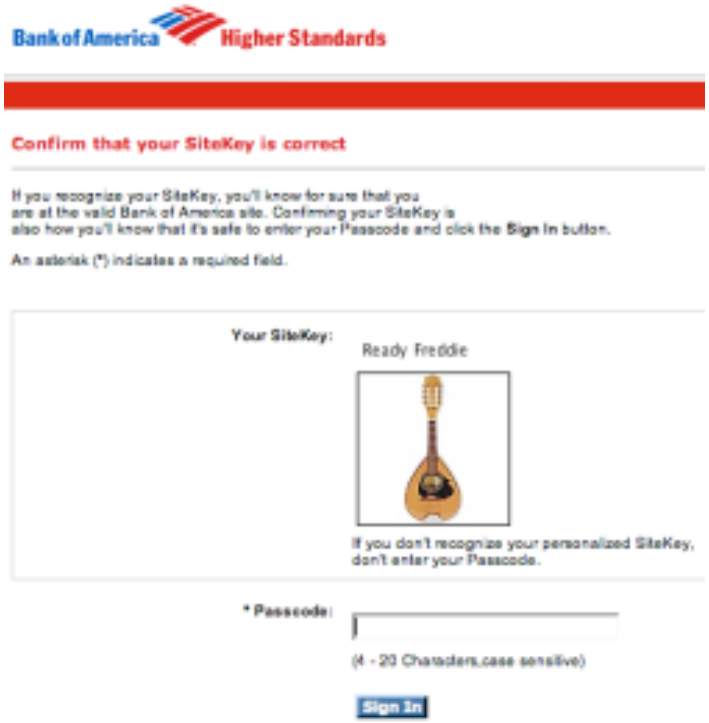
    $$Sign_A(M) := M, \{(M)\}SK_A$$
  - Since only Alice knows SK_A only she can perform this operation
  - On the other hand, anyone knowing Alice's public key PK_A can apply this to the second term $\{(M)\}SK_A$ and check that this agrees with the first, i.e. M

Outline

- Product Authentication
- Message Authentication (Integrity Verification)
- Server Authentication
- Software Identification (Fingerprinting)
- Authentication Protocols in Distributed Systems (Beyond User Authentication)
- Humanness Authentication (CAPTCHA)
- Summary

A naïve method

- Personalised secret shared between the user and the server (used by some banks)
  - A word or a sentence
  - An image (e.g., SiteKey used by Bank of America)
  - …



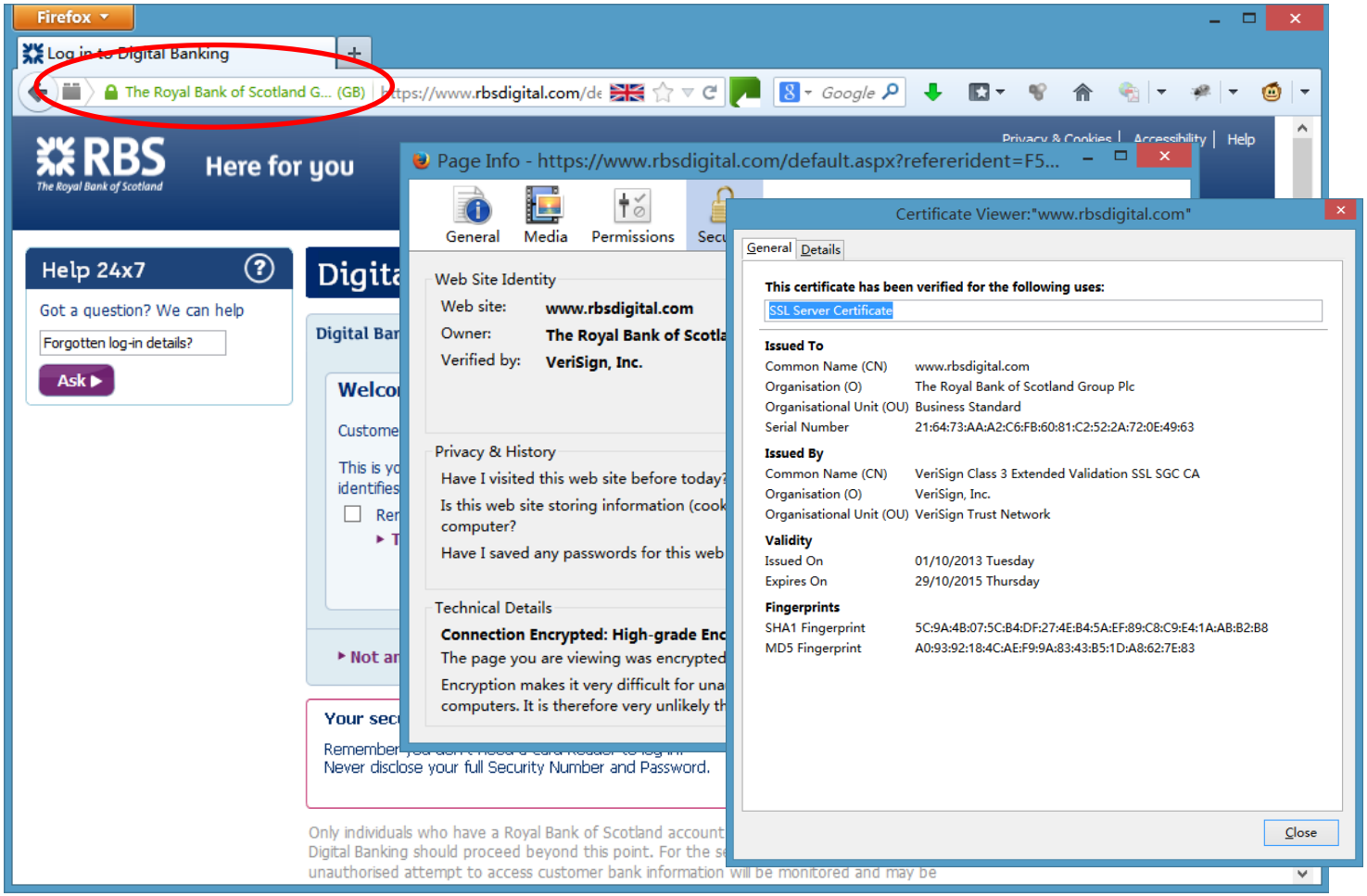
Security protocols

- Protocols allowing server authentication
  - Typically rely on a trusted third party (e.g., a certificate authority)
  - Example: server's digital certificate used with TLS
- Protocols maintaining confidentiality if the server is not the genuine one
  - Example: using the public key of a known server to encrypt all communications
  - Implicit server authentication: no actual authentication, but only the genuine server can decrypt the communications.
- Mutual authentication
  - The client authenticates the server + The server authenticates the client.
  - Example: client certificate + server certificate over TLS

Website authentication with server certificate (and HTTPS)

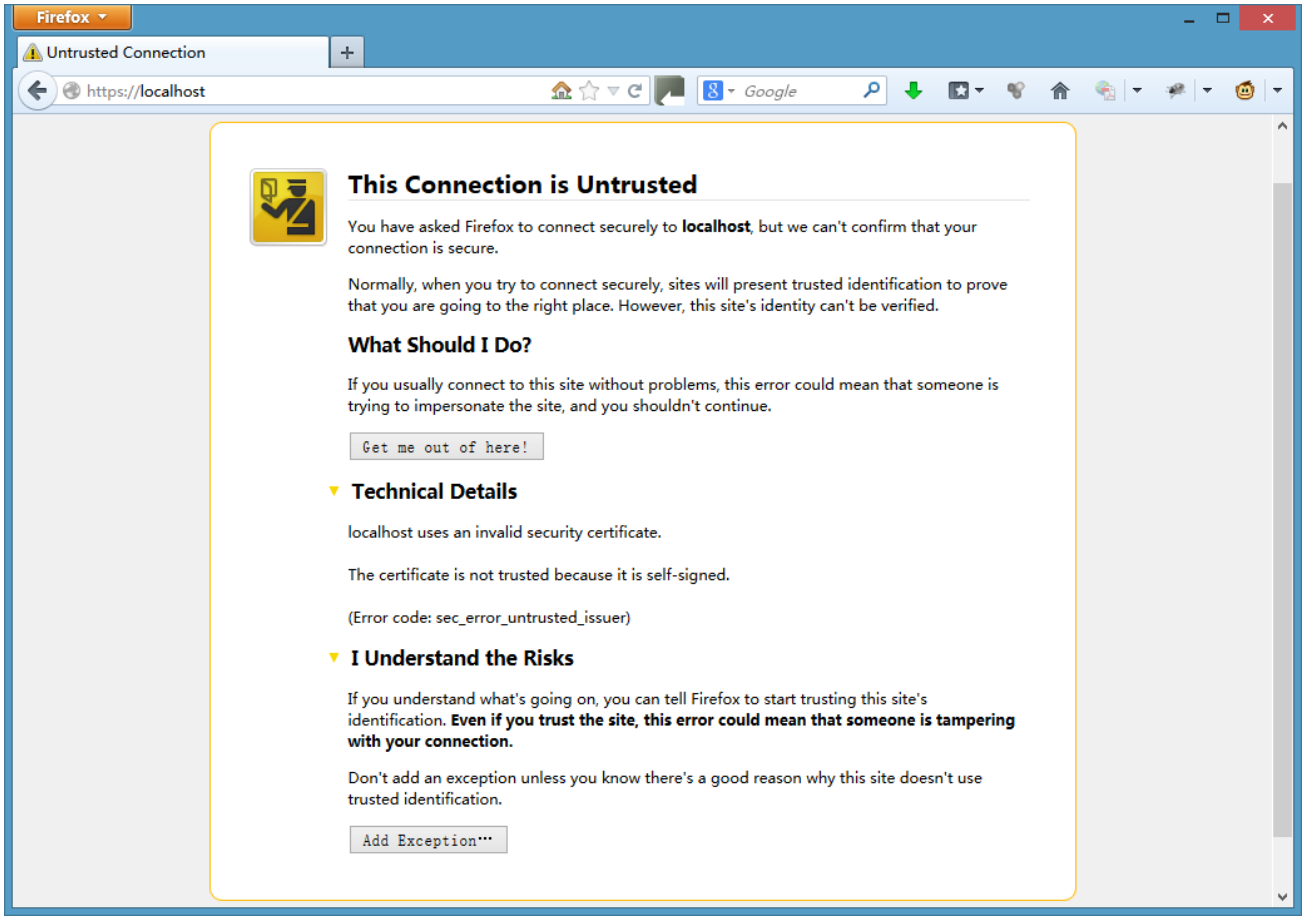
University of Kent

- Do you **often** check the content of a website's digital certificate's against the claimed owner?



Do you trust **problematic** CAs?

- Have you seen a web browser warning about a digital certificate used by a website like the following one?



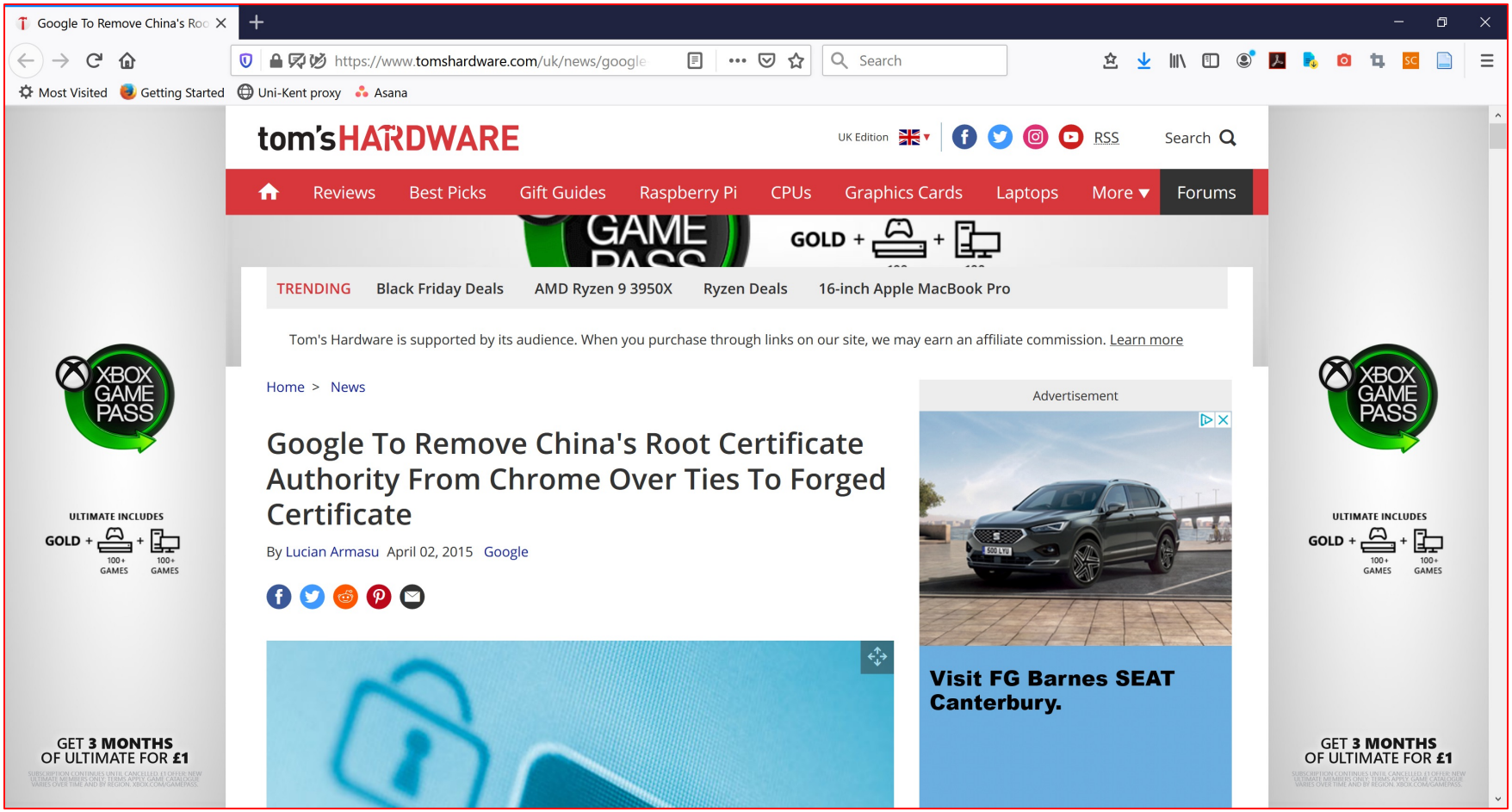
Do you trust **problematic** CAs?

University of Kent

- **<u>If YES to the previous question</u>**: Did you choose to ignore the web browser warning(s) because you felt you could trust the website(s) you were visiting?

Should you trust CAs?

- What if a CA is (or was) actually **not** trustworthy?



“In-class” exercise

- Think about all digital certificate warnings you’ve ever encountered when using a web browser.
  - If you haven’t encountered many, consider what you may come across.
- List all different types of warnings.
- Which ones are more problematic?

- If CAs cannot be trusted, do we have other better options?

Outline

- Product Authentication
- Message Authentication (Integrity Verification)
- Server Authentication
- Software Identification (Fingerprinting)
- Authentication Protocols in Distributed Systems (Beyond User Authentication)
- Humanness Authentication (CAPTCHA)
- Summary

Software fingerprinting

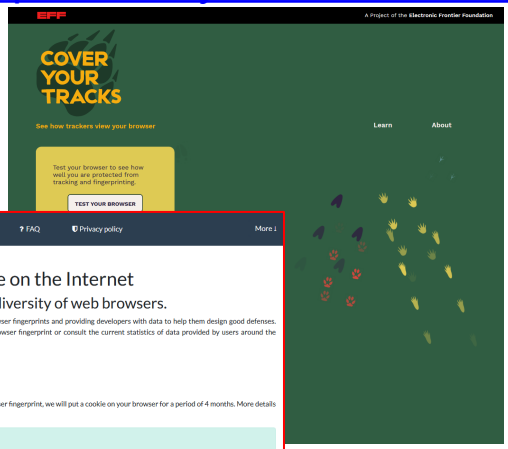
- Software tools can have unique identifiable behaviours.
- Some may be very stable so can be captured to identify unknown software.
- Similar to device identification.

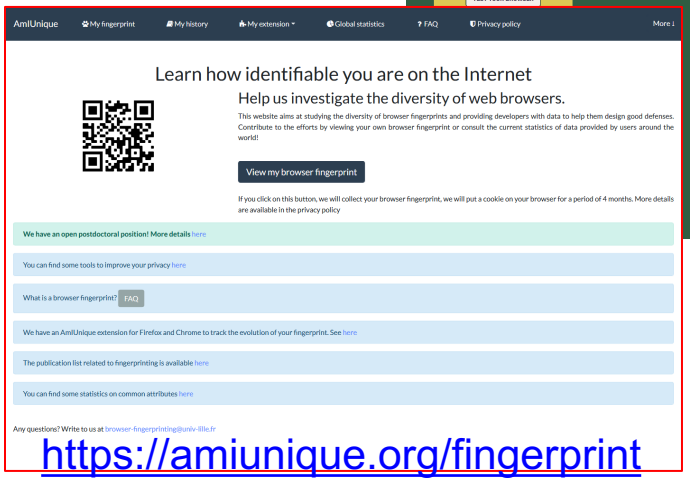
- When malware is the one to identify
  - ⇒ Malware detection
  - E.g., ransomware deployment detection
    - Gavin Hull, Henna John, Budi Arief, “Ransomware Deployment Methods and Analysis: Views from a Predictive Model and Human Responses”, *Crime Science* 8(2), 2019. https://rdcu.be/bmtVa

Example: Web browser fingerprinting

- Web browser is a piece of software.
- It has some unique behaviours and often leaks information about the user to the server ⇒ Privacy concerns
  - Even when private browsing mode is used.
- What information can be used for fingerprinting?
  - Cookies, IP addresses, …
  - Operating system and version
  - Web browser (user agent) name and version
  - Screen resolution
  - Time zone
  - Language
  - Font face and font size
  - Active background color
  - …

https://coveryourtracks.eff.org/





https://amiunique.org/fingerprint

Outline

- Product Authentication
- Message Authentication (Integrity Verification)
- Server Authentication
- Software Identification (Fingerprinting)
- Authentication Protocols in Distributed Systems (Beyond User Authentication)
- Humanness Authentication (CAPTCHA)
- Summary

From my slides (2019-2020): Protocols

Protocol and Cryptographic Protocol

- Protocol is:
  - A series of steps
  - Involving two or more parties
  - Designed to accomplish a task
- Other characteristics of a protocol:
  - Everyone involved must know it and all of the steps to follow in advance
  - Everyone involved must agree to follow it
  - It must be unambiguous
  - It must be complete

From my slides (2019-2020): Protocols

Protocol and Cryptographic Protocol (contd.)

- Cryptographic Protocol is essentially a protocol that uses cryptography
  - Uses cryptographic mechanisms/primitives
  - The parties might be friends or adversaries
  - It involves some cryptographic algorithm, but the goal is generally beyond simple secrecy
    - Identity confirmation, contract signing, collaborative computation, ...
  - Typically, the main aims are to prevent or detect eavesdropping or cheating
    - It should not be possible to do or learn more than what is specified in the protocol

From my slides (2019-2020): Protocols

Protocol and Cryptographic Protocol (contd.)

- Started with the advent highly distributed applications
  - Really took off with the invention of Public Key techniques
- Some basic security functions can be provided purely by classical crypto
  - e.g. authentication and secrecy
  - But still face the key distribution problem
- Other functions, e.g., non-repudiation, really not possible without Public Key
  - Perhaps with tamper resistant/evident devices

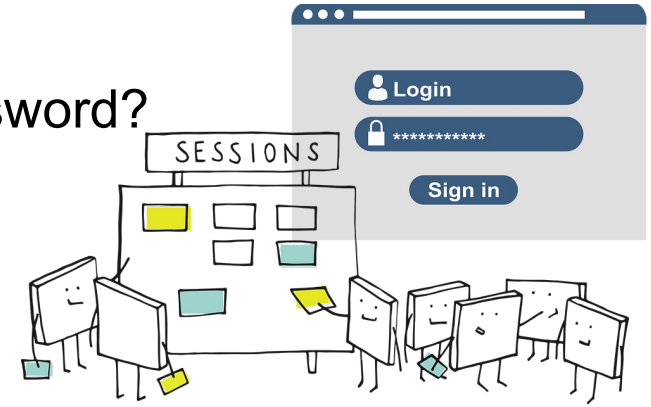
From my slides (2019-2020): Protocols

Challenges

- Designing a good cryptographic protocols is remarkably difficult!
  - Even specifying the requirements precisely is non-trivial
  - Complex, distributed and hostile environment
  - Modelling the adversary capabilities and anticipating attack strategies is challenging too:
    - We need to model the adversary's powers of observation, computation and deduction
  - Highly concurrent systems, typically unbounded
  - Has been a dynamic area of research for decades and still going strong

Authentication, protocol and sessions

- User authentication as a **challenge-response protocol**
  - The verifier issues a challenge; the prover responds with a proof.
  - Password as an example
    - The challenge: What is the password?
    - The response: password123



- Users and sessions
  - A session: from login to logout
  - An **authentication token** may be generated to indicate the session (NOT the user) has passed authentication.
    - Example: a session variable in a web browser session
  - A **session key** can be used to protect communications of a session.
    - Example: the key used for protecting an HTTPS session

Key establishment/exchange protocols

- Need to involve at least two remote partners (prover and verifier), and often involve a (trusted) third party (e.g., an identity provider, or a CA).
  - $\Rightarrow$ Allows mutual authentication!
- Normally involves multiple rounds.
- Normally involves use of cryptographic nonces ("number used once") to prevent **replay attacks**.
- Based on cryptographic primitives

Needham-Schroeder protocol (1978)

- Assumptions:
  - A and B both trust a server S (an identity provider).
  - A and S share a secret key K_{AS}.
  - B and S share a secret key K_{BS}.
- The goal
  - For A and B to establish a session key K_{AB}.
- The protocol
  1. A → S: A, B, N_A
  2. S → A: $\{N_A, B, K_{AB}, \{K_{AB}, A\}_{K_{BS}}\}_{K_{AS}}$
  3. A → B: $\{K_{AB}, A\}_{K_{BS}}$
  4. B → A: $\{N_B\}_{K_{AB}}$
  5. A → B: $\{N_B - 1\}_{K_{AB}}$

Symmetric-key version

***N_A* is a nonce.**

$\{x\}_K$ = using *K* as the key to encrypt *x*

***N_B* is also a nonce.**

Needham-Schroeder protocol: an attack

- Assumptions
  - E (an attacker) somehow got K_{AB}.
    - How?
  - E observed the session between A and B when K_{AB} was established.
- The goal
  - E wants to reuse K_{AB} to impersonate A to communicate with B in future.
- The (replay) attack

3. E → B: $\{K_{AB}, A\}_{K_{BS}}$
4. B → E: $\{N_B\}_{K_{AB}}$
5. E → B: $\{N_B - 1\}_{K_{AB}}$

***B* will believe he is communicating with A after this step.**

Fixing Needham-Schroeder protocol

- Introducing a **timestamp** so old session keys cannot be simply reused.
  - This is a common security fix for many security protocols!
- Adding authentication of A to the protocol.
  1. A → B: A
  2. B → A: $\{A, N_B'\}_{K_{BS}}$
  3. A → S: $A, B, N_A, \{A, N_B'\}_{K_{BS}}$
  4. S → A: $\{N_A, B, K_{AB}, \{K_{AB}, A, N_B'\}_{K_{BS}}\}_{K_{AS}}$
  5. A → B: $\{K_{AB}, A, N_B'\}_{K_{BS}}$
  6. B → A: $\{N_B'\}_{K_{AB}}$
  7. A → B: $\{N_B' - 1\}_{K_{AB}}$

(A, N_B') binds the nonce N_B' with the sender A.

Kerberos

- Developed at the Massachusetts Institute of Technology (MIT) for Project Athena in the 1980s
- Became an IETF standard in 1993 (RFC 1510)
  - A clearer and more detailed description in 2015 (RFC 4120)
- Widely used by many real-world systems
- The protocol is a bit more involved, so I will let you to self-study it if you are interested.
  - Use an improved version of Needham-Schroeder protocol with timestamps.
  - Secure against eavesdropping and replay attacks

Outline

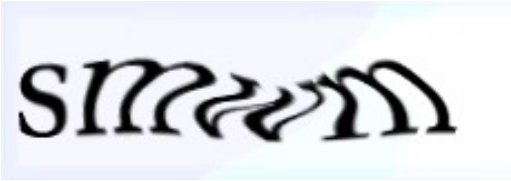
- Product Authentication
- Message Authentication (Integrity Verification)
- Server Authentication
- Software Identification (Fingerprinting)
- Authentication Protocols in Distributed Systems (Beyond User Authentication)
- Humanness Authentication (CAPTCHA)
- Summary

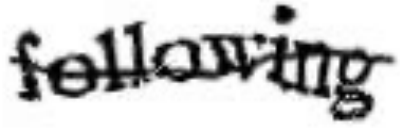
CAPTCHA: Humanness authentication

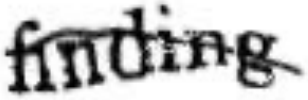
- CAPTCHA = **C**ompletely **A**utomated **P**ublic **T**uring test to tell **C**omputers and **H**umans **A**part
- A computer verifiers a claimant's **humanness**.
- Why do we need CAPTCHA?
  - Originally, to stop spammers from using a script for creating email accounts on free email services.
  - Now it is used for all sorts of tasks where a website needs to ensure that it is dealing with a human.

CAPTCHA: Humanness authentication

- How does CAPTCHA work?
  - The program generates some random text.
  - It then produces a distorted version of this text that the user must decipher.
  - Easy for human to solve, but hard for computer.





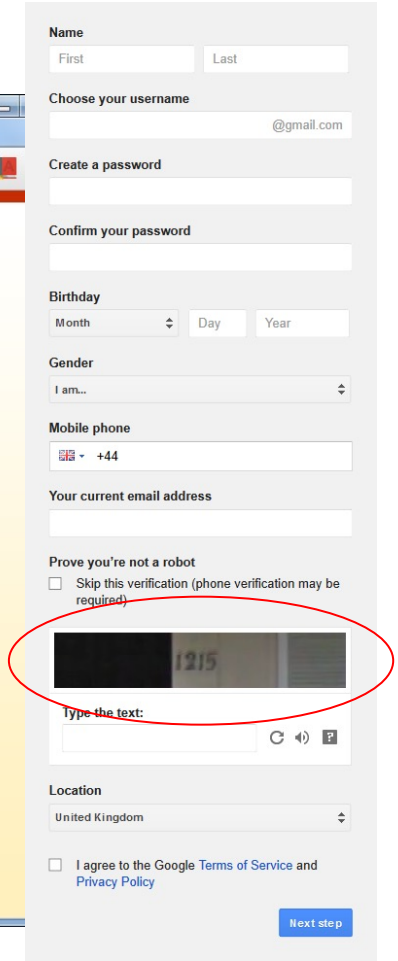


CAPTCHA: Humanness authentication

- Some variants:
  - Voice-based
  - Image-based
  - reCAPTCHA

reCAPTCHA v1: Not 100% CAPTCHA

- Google **reCAPTCHA** was used for recognising words in digitised books for Google Books and street addresses in Street Views for Google Maps.

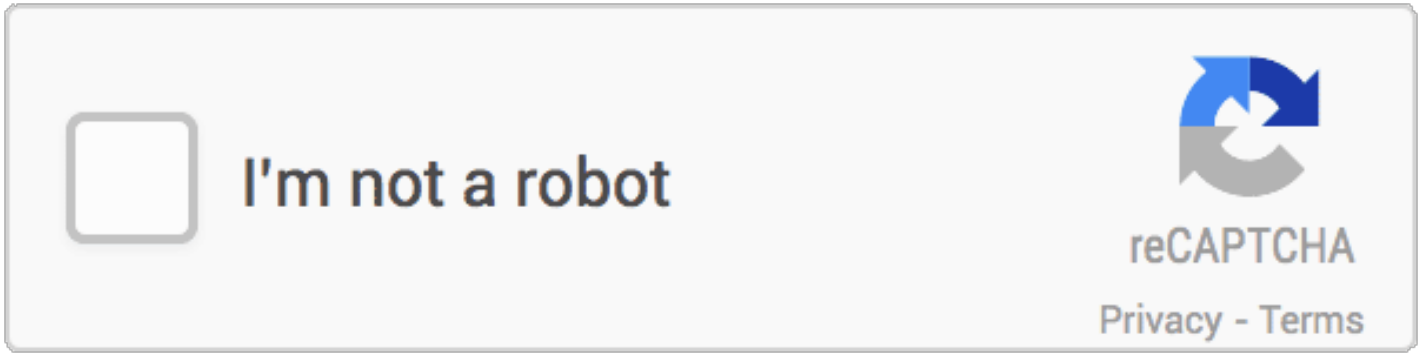


https://www.google.com/recaptcha/about/

reCAPTCHA v2

- Invisible CAPTCHA
  - Risk-based humanness authentication
  - A visible CAPTCHA will pop up when the risk is considered high.

- Checkbox “CAPTCHA”
  - A (mouse) behaviour-based CAPTCHA



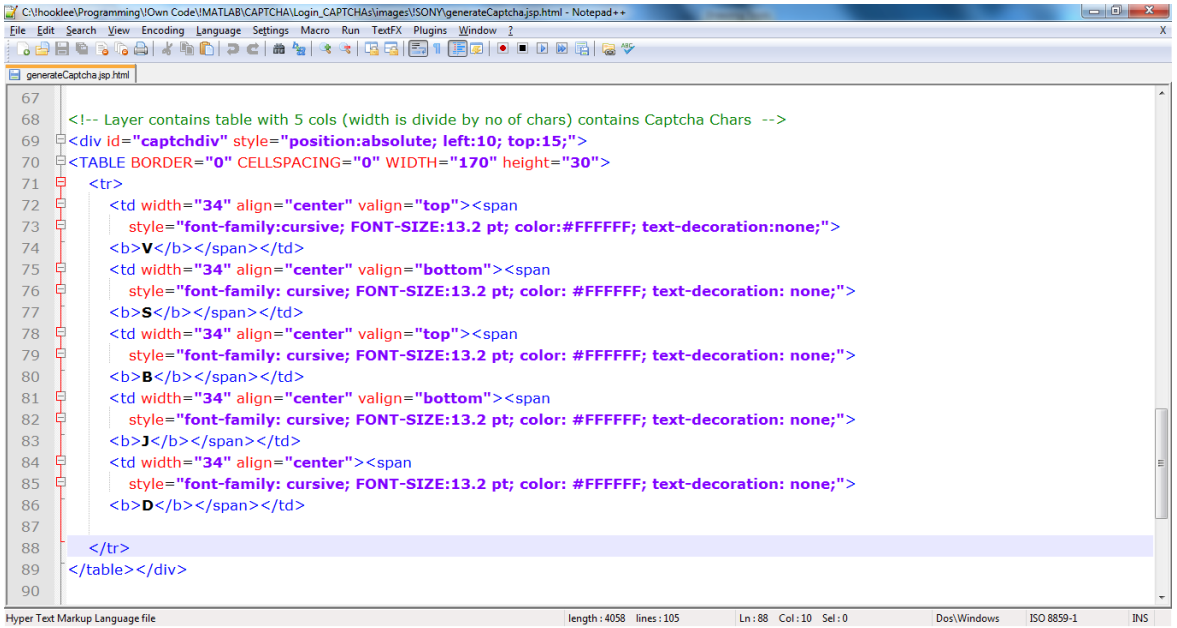
Bad CAPTCHA

- CAPTCHA @ SONY web forum (2011)
  - In Google Chrome 21.0.1180.75 m:

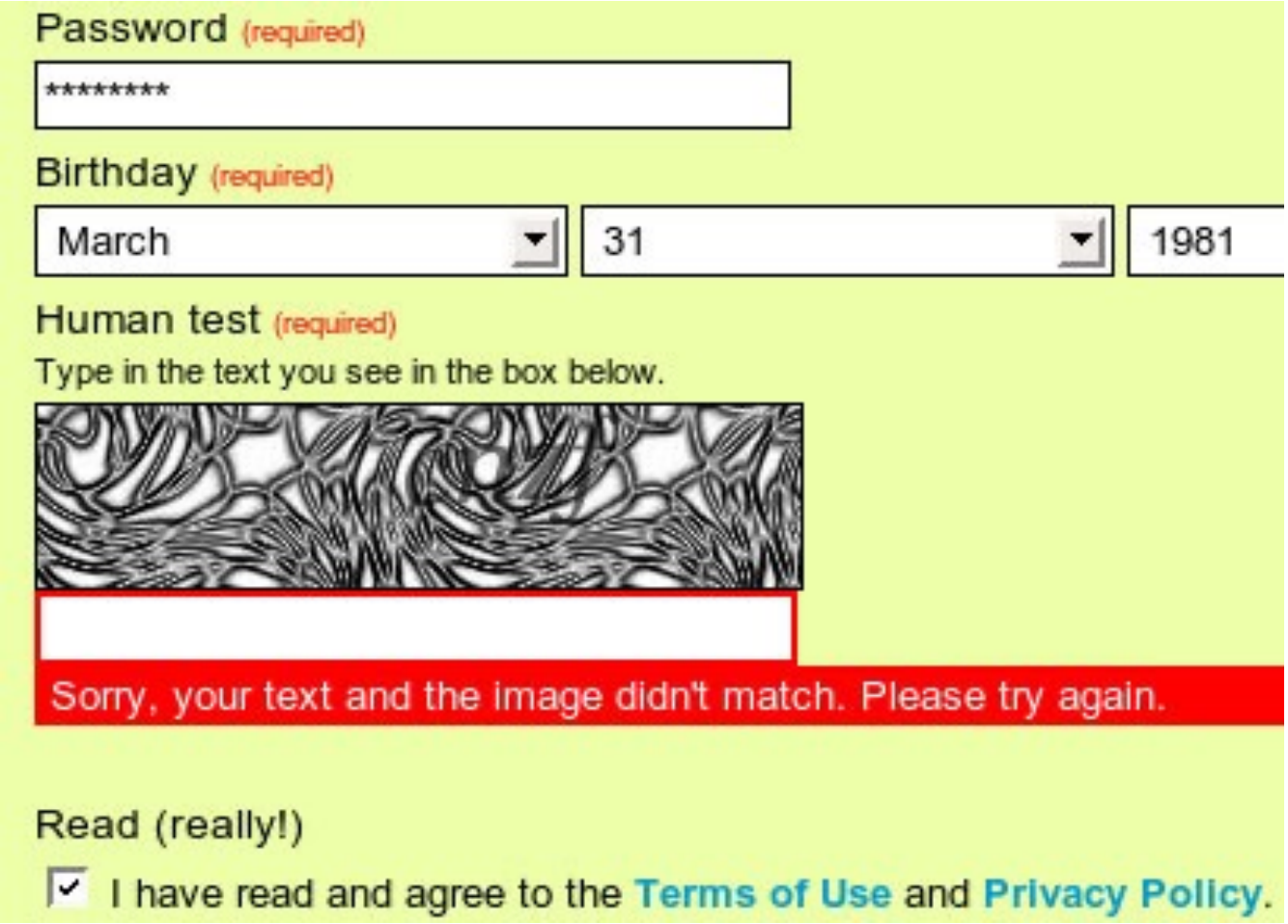
  - In Mozilla Firefox 15.0.1:

  - In MSIE 9.0.8.112.16421:

  - It is obviously weak, but…



Some are very hard to answer correctly…



From: http://www.seosmarty.com/impossible-captcha-it-doesnt-really-matter-if-you-are-human-or-not/

… or just impossible!*

Qualifying question

Just to prove you are a human, please answer the following math challenge.

Q: Calculate:

$$\frac{\partial}{\partial x}\left[6\cdot\sin\left(x-\frac{\pi}{2}\right)+3\cdot\cos\left(2\cdot x-\frac{\pi}{2}\right)\right]\Bigg|_{x=\pi}.$$

A:

mandatory

Note: If you do not know the answer to this question, reload the page and you'll (probably) get another, easier, question.

From: http://www.seosmarty.com/impossible-captcha-it-doesnt-really-matter-if-you-are-human-or-not/

* Unless you are a robot, ironically!

CAPTCHA + _____ authentication

- CAPTCHA + Passwords
  - Use as a way to prevent online attacks
  - Use as a way to enter a password

Bin B. Zhu, Jeff Yan, Guanbo Bao, Maowei Yang, and Ning Xu. "Captcha as graphical passwords—A new security primitive based on hard AI problems." *IEEE Transactions on Information Forensics and Security* 9(6):891-904, 2014. https://ieeexplore.ieee.org/document/6775249

- CAPTCHA + Biometrics
  - Used as a way for liveness detection

Erkam Uzun, Simon Pak Ho Chung, Irfan Essa, and Wenke Lee. "*rt*Captcha: A Real-Time CAPTCHA Based Liveness Detection System." 2018 *Network and Distributed Systems Security (NDSS) Symposium*, 2018. http://dx.doi.org/10.14722/ndss.2018.23253

- …

If you are interested in CAPTCHA further

University of Kent

- http://www.hooklee.com/CAPTCHA.asp
- Antreas Dionysiou and Elias Athanasopoulos, "SoK: Machine vs. machine – A systematic classification of automated machine learning-based CAPTCHA solvers," *Computers & Security*, vol. 97, Article No. 101947, Elsevier, 2020
- José María Gómez Hidalgo and Gonzalo Alvarez, "CAPTCHAs: An Artificial Intelligence Application to Web Security," *Advances in Computers*, vol. 83, Chapter 3, pp. 109-181, Elsevier, 2011
- Marti Motoyama, Kirill Levchenko, Chris Kanich, Damon McCoy, Geoffrey M. Voelker and Stefan Savage, "Re: CAPTCHAs – Understanding CAPTCHA-Solving Services in an Economic Context," in *Proceedings of 19th USENIX Security Symposium*, USENIX, 2010
- Elie Bursztein, Steven Bethard, Celine Fabry, John C. Mitchell, Dan Jurafsky, "How Good Are Humans at Solving CAPTCHAs? A Large Scale Evaluation," in *Proceedings of the 2010 IEEE Symposium on Security and Privacy*, pp. 399-413, IEEE Computer Society, 2010

Outline

- Product Authentication
- Message Authentication (Integrity Verification)
- Server Authentication
- Software Identification (Fingerprinting)
- Authentication Protocols in Distributed Systems (Beyond User Authentication)
- Humanness Authentication (CAPTCHA)
- Summary

Summary

- Apart from user authentication, there are many other types of authentication
  - Product, message, server, software, device, …
  - Even “humanness” has to be authenticated sometimes!
- We also looked briefly at (cryptographic) protocols
  - Particularly useful in distributed systems
  - Sometimes these protocols are vulnerable to attacks
  - Designing and implementing secure protocols is not easy!
- Next lectures: access control and authorisation