Week 7

Crypto Protocols and Network Security Overview

THE UNIVERSITY OF SYDNEY

Secret Splitting

- Problem: You are the CEO of Coca-Cola. You're responsible for keeping the formula secret from Pepsi's industrial spies. You could tell your most trusted employees, but…
  - They could defect to the opposition.
  - They could be threatened and coerced.

- How can a secret be split among multiple parties such that each piece by itself is useless?

Secret Splitting with XOR

- Suppose Trent wants to protect the message m:
  - 1. Generate a random string r, the same length as m.
  - 2. Compute $s = m \oplus r$.
  - 3. Give Alice r, and give Bob s.

- Each piece r, s is called a **shadow** of the message m. To reconstruct m, Alice and Bob can XOR their shadows together:

$$s \oplus r = m$$

- If r is truly random, the system is perfectly secure (similar to One Time Pad).
- The scheme may be extended to n people by generating $n - 1$ random strings $r_1, \ldots, r_{n-1}$. Give the first person r_1, the second person r_2, and so on up to r_{n-1}, and give the n^{th} person $r_1 \oplus \cdots \oplus r_{n-1} \oplus m$.

Secret Splitting with XOR

- Secret splitting aims to enhance reliability without increasing risk through distributing trust.

- Issues with secret splitting with XOR:
  - The system is adjudicated by Trent.
    - Trent can hand out rubbish and say it is the secret.
    - Trent can say he is splitting a secret 4 ways but only splitting it between the first two people.
  - All parties know the length of the message.
  - The message is malleable: by flipping bits in any part, the recovered message changes.
  - All parties are required to recover the message.

Secret Sharing

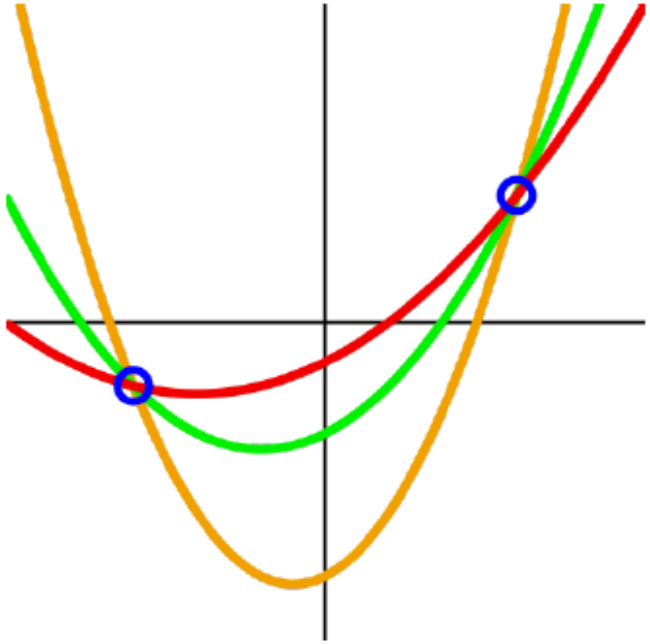
- Problem: You are responsible for the funds of a company.
  - Ensure that no single executive can withdraw funds.
  - Ensure that no pair of executives can withdraw funds.
  - You want at least three of five executives to agree before initiating a fund withdrawal. This is called a (3, 5)-threshold scheme.

- Shamir's Secret Sharing is an algorithm for dividing a secret into m pieces, where only n of them are required to reconstruct the original secret.

Shamir's Secret Sharing (SSS)

- A polynomial of degree n can be uniquely defined by plotting $n + 1$ points on that polynomial.

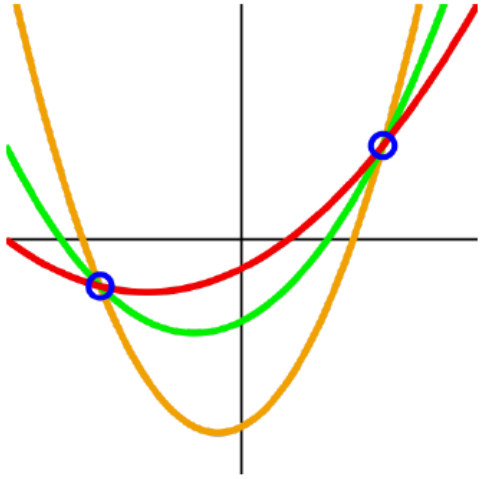
- Example: $y = ax^2 + bx + c$
  - Infinite possibilities for the coefficients a, b, c with only 2 points.
  - Coefficients are uniquely determined with 3 points.



Shamir's Secret Sharing (SSS)

- Trent wishes to distribute a message m amongst n users, where any group of t users ($1 \leq t \leq n$) can recover m.
  - 1. Choose a prime $p > \max\{m, n\}$.
  - 2. Create the polynomial $f(x) = m + a_1 x + a_2 x^2 + \cdots + a_{t-1} x^{t-1}$, where the coefficients $0 \leq a_i < p$ are random and independent.
  - 3. Trent selects n distinct points x_i, with $1 \leqslant x_i < p$.
  - 4. Trent gives $(x_i, f(x_i))$ to person i.

- Once t people pool their $(x_i, f(x_i))$ points, then the polynomial $f(x)$ can be reconstructed, and m recovered.



Commitment Protocols

THE UNIVERSITY OF SYDNEY

Commitment Schemes (Bit Commitment Schemes)

- Problem: Alice wants to send a secret message to Bob which can only be revealed by Bob at a later stage with further information from Alice. Bob can check that the message has not been changed.

- Commitment:
  - 1. Bob generates a random string r, and sends it to Alice.
  - 2. Alice generates a random key k, and sends Bob encryption $E_k(r \parallel m)$.
- Revelation:
  - 1. Alice sends Bob the key k.
  - 2. Bob decrypts the message with k, and verifies r in commitment.

- Discussion:
  - r is needed for freshness, and to stop Alice from finding colliding messages, with $E_k(m) = E_{k'}(m_2)$. She needs to commit to m at the time she sends $E_k(r \parallel m)$.
  - Bob does not know k until revelation.

Non-interactive Bit Commitment

- Commitment:
  - 1. Alice generates random strings r, s, and computes $x = h(r \parallel s \parallel m)$ (aka blob) with secret message m.
  - 2. Alice sends Bob (r, x).
- Revelation:
  - 1. Alice sends Bob the remaining data of (s, m).
  - 2. Bob verifies that $h(r \parallel s \parallel m)$ is the same as the x he received.

- Discussion:
  - Bob does not have to send any messages.
  - Alice sends a message to commit, and a message to reveal.
  - Since h is a crypto hash function, Alice cannot find t such that $h(r \parallel s \parallel m) = h(r \parallel t \parallel m)$.
  - s is kept secret so that Bob can't brute force the message space, before Alice sends m.

Fair Coin Flipping using Commitment Schemes

- Problem: Alice and Bob want to decide who will play first in a game of online chess. They want to flip a coin to resolve the situation.
  - Alice doesn't trust Bob to flip the coin.
  - Bob doesn't trust Alice to flip the coin.

- How can a coin be flipped fairly?
  - 1. Alice commits to a bit b using a commitment scheme.
  - 2. Bob tries to guess the bit.
  - 3. Alice reveals the bit: if Bob guessed correctly, he wins the toss. Otherwise, Alice does.

- Discussion:
  - The security of this algorithm lies in the security of the commitment scheme. In particular, the blob of the commitment scheme should not give away anything about the message inside (such as low-order bits).

Fair Coin Flipping using Public Keys

- We require a commutative public key cryptosystem so that
$$E_A(E_B(m)) = E_B(E_A(m))$$
- To perform a fair coin flip:
  - 1. Alice and Bob generate commutative public key pairs A, B respectively.
  - 2. Alice generates two secret random numbers rT and rH.
  - 3. Alice sends Bob encrypted messages $E_A(\text{heads}, rH)$ and $E_A(\text{tails}, rT)$ in a random order.
  - 4. Bob selects one of Alice's messages, call it $E_A(x)$, and sends Alice encrypted message $E_B(E_A(x))$.
  - 5. Alice decrypts Bob's message and sends back the result $E_B(x)$.
  - 6. Now Bob can decrypt $E_B(x)$ and see what he has selected, either (heads, rH) or (tails, rT). Then Bob sends the result back to Alice.
  - 7. Alice verifies that Bob's response matches up with her rT or rH.

Fair Coin Flipping using Public Keys

- Discussion:
  - The algorithm is self-enforcing: either party can detect the other cheating, without requiring a trusted third party.
  - Bob learns the result of the coin flip **before** Alice. He can't change the result, but he may delay it ("flipping the coin into a well").
  - Coin flipping has use in session key generation, as neither party can influence the result of each flip. For example, in Diffie-Hellman, one party selects an exponent after the first.

Mental Poker

- How to fairly deal cards to Alice and Bob using commutative key pairs?
  - 1. Bob encrypts 52 messages: (two of clubs), … , (ace of spades), using his secret key and sends them all to Alice in random order (shuffled).
  - 2. Alice picks 5 encrypted messages randomly and sends them to Bob. Bob decrypts the messages: This is his hand.
  - 3. Alice picks another 5 encrypted messages and encrypts them with her secret key before sending them to Bob. Bob decrypts the messages and sends them back to Alice. Alice decrypts the messages: This is her hand. (Repeat steps 2 and 3 for further cards)
  - 4. At the end of the game, Alice and Bob reveal their secret keys to ensure no-one cheated.

- Further reading: A. Shamir, R. Rivest, and L. Adleman, "Mental Poker", Technical Memo LCS/TM-125, Massachusetts Institute of Technology, April 1979. https://apps.dtic.mil/dtic/tr/fulltext/u2/a066331.pdf

Covert Channels and Steganography

THE UNIVERSITY OF SYDNEY

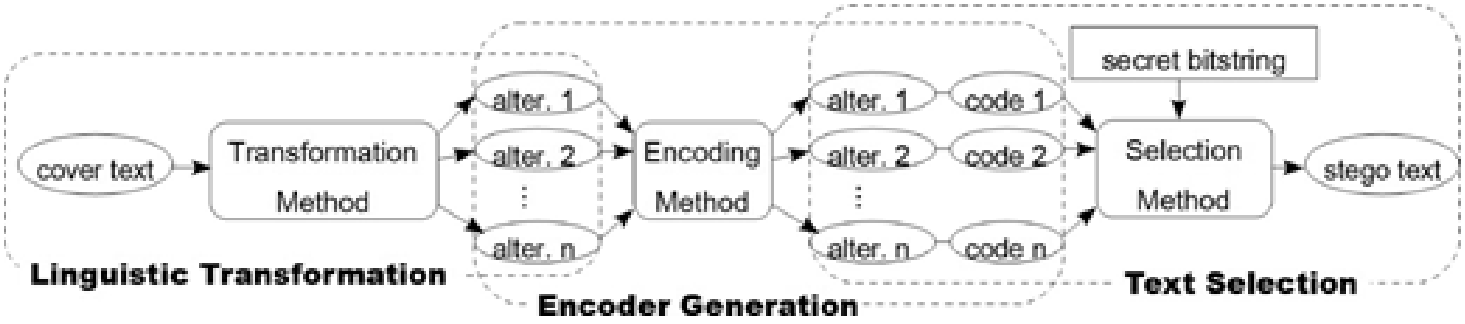
Covert Channels

- Alice and Bob want to set up a covert (or subliminal) channel to communicate secretly in the presence of a warden, Walter, who is monitoring the communications channel. Walter can see their messages and might attempt to deceive them by planting false messages.

- Simplest level:
  - Alice and Bob can use **steganography** (information hiding) to send hidden messages in places no-ones suspects a message exists.
    - Steganography is not cryptography: it's "security through obscurity".
    - Steganography is usually used together with cryptography.

Text Steganography

- Solutions to send bits using text:
  - Calculate: count(words in sentence): If even → 0; If odd → 1
  - Synonym substitution: ~0.67 bits per sentence. [Topkara et al.]
  - Syntactic substitution: ~0.5 bits per sentence. [Atallah et al.]
  - Contextual Synonym Substitution and Vertex Coding: ~1 bit per sentence. [Chang & Clark]



Word	Value
Complete	0
Finish	1

Word	Value
Labor	00
Project	01
Task	10
Undertaking	11

"He **finished** the **project**"
⇓
encode 010
⇓
"He **completed** the **task**"

Image Steganography

- Imagine a grayscale image of 32x32 pixels, where each pixel is an 8-bit value (0-255).
  - One pixel would contain the bits:

128s	64s	32s	16s	8s	4s	2s	1s

  - If we are happy with a slight trade-off for quality, we can afford to lose the least-significant bit:

128s	64s	32s	16s	8s	4s	2s	msg

- The changes to the image will be imperceptible to a casual observer. We can now encode a secret message of 32x32 =1024 bits (128 characters)!

Image Steganography

- If we are using a colour image (RGB), we can use 3 bits per pixel. Doubling the resolution quadruples the amount of data that can be hidden.
- Settle for some lost quality, and we can even go to 6 bits per pixel (2 bits per colour).
- Techniques exist that work with lossy recompression and even resizing.

Take the two lowest bits

→

Network Steganography

- Loki: Bidirectional covert UNIX shell client using the data field in ICMP type 0 (Echo Reply) and type 8 (Echo Request) packets.
- ICMP Backdoor: Reusable tunnel library Messages fragmented to look more like ping packets (multiples of 64 bytes)
- Rwwwshell: Backdoor emits requests as HTTP Response packets Output from commands return from the slave as cgi script HTTP GETs
- AckCmd: A command shell sending all traffic via TCP ACK packets.
  - Only uses allowing clients to contact servers through some firewalls

Network Security Overview

THE UNIVERSITY OF SYDNEY

Network Security

- Internet security vendors offer a wide range of products to help keep you secure from security attacks, e.g., Network Scanning Tools, Firewalls, Virtual Private Networks (VPN), Intrusion Detection Systems (IDS), Public Key Infrastructure (PKI), Biometrics, etc.

- The products have limitations:
  - Hackers also run network scanning tools
  - Firewalls are often “walls” having holes in them
  - Virtual Private Networks work on top of the Internet
  - Intrusion Detection Systems work well on old attacks, not newer ones
  - Public Key Infrastructure requires complex infrastructure and management
  - Biometrics can be fooled and cannot revoke or issue new keys

Challenges of Network Security

- The Internet is not secure by design with many vulnerabilities
  - Operating systems
  - The IP stack and 802.11/WEP
  - User protocols such as Telnet, FTP, RSH and HTTP
  - Network protocols such BGP and DNS
  - Network management protocols such as SNMP
  - Security features of programming languages such as C
  - Lack of proper infrastructure
  - Lack of quality developers
  - Poor design and programming practice
  - (design choices, implementation, assumptions, ...)

Distributed Denial of Service (DDOS) Attacks

- July 1999: The Computer Emergency Response Team (CERT) issues an advisory on Denial-of-Service attacks
- Sep 1999: Packet Storm receives first copies of DDoS tools
- Nov 1999: CERT warns of new class of attacks (DDoS) and tools in circulation at CISAC Information Warfare conference
- Dec 1999: Packet Storm release new tools and launches Storm Chaser 2000: Next Generation CyberDefence.
- Feb 7 2000: Yahoo 3hr outage
- Feb 8 2000: Ebay 5hr outage; Amazon 3:45hr outage; CNN 3:30hr outage
- Feb 9 2000: ZDnet 3:15hr outage; E*trade 2:45hr outage

- The attack: An amplified denial-of-service attack on the routers connecting these websites to the Internet (Amplified Ping and SYN floods)
- Further reading: https://blog.cloudflare.com/ddos-attacks-on-australian-universities/

Botnet Attacks

- Conficker Worm (2008) targeted Microsoft Windows and created a botnet which had more computing power than any existing supercomputer. It could "fight back" by creating multiple variations that repaired previous vulnerabilities and became more difficult to stop

- Conficker was first released three weeks after a fix was released to the public. In a perfect world, how could a patched vulnerability cause an issue?
  - October 23, 2008: The patch for that issue (MS08-067) was released
  - January 2009: 30% (or more) of Windows PCs remained unpatched
  - The vulnerability hit Windows 2000, Windows XP, Windows Vista, Windows Server 2003, Windows Server 2008.
  - https://www.cisa.gov/news-events/alerts/2009/03/29/conficker-worm-targets-microsoft-windows-systems

Economies of the Internet

- Organisations need to understand that the Internet brings numerous advantages but equally large disadvantages as well
  - Information leakage (US govt documents → Wikileaks)
  - Operationally exposing your internals to the world 24x7x365
  - Increased risk associated with increased chance of compromise
  - Ease at which attackers can launch attacks and get away with them
- There is no single Internet Police and no international jurisdiction

Internet Security Vulnerabilities

- Security is always catching-up
  - Always a significant time delay between finding, reporting, advising and fixing problems
- Security is usually reactive
- Security is perceived as a cost center, not a profit center
- Homogenous nature of the Internet
- Heterogeneous nature of the Internet
- Political issues, export restrictions
- Patents
- Humans using the Internet

The Internet is a Monoculture

- Throughout Internet, many tools have huge market dominance. e.g., OpenSSL; computers running Windows; nameservers running Bind; web servers running Apache or Nginx; etc.

- Software is almost always an exact copy, across different version. If you can break into one version, you can break into them all. Thus, an attack against any of these tools will result in numerous owned machines.

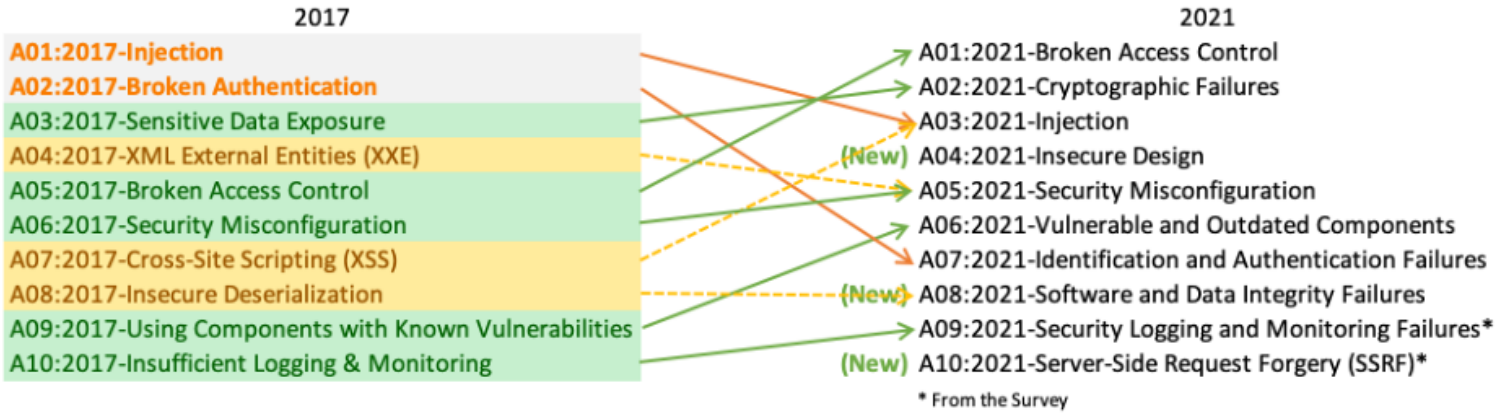
- IoT: More and more devices (smart phones, tablets, watches, pacemakers, fridges) are coming online where security is not even on the feature list.

Common Beliefs about Security

- The common security philosophy is that if you secure the perimeter, you can keep the insides soft and gooey (marshmallow). This has always been a very bad assumption.
- Nowadays it is even worse:
  - There is no clear border for your network.
  - You cannot trust anyone.
  - There are simply too many ways into your network:
  - Internet connections (T1, cable, ADSL, frame relay …)
  - Every machine, including those ones set up by an intern four years ago running a system or tool that no-one has updated since
  - 802.11 wireless networks (the record is well over 15 kilometres with a good antenna and amplifier)
  - Third party connections (vendors, partners, clients …)
- Users are the 90% of the problem and they are already inside!

Security Flaws

- Modern security still has all the problems of non-modern security: the only difference is technology is more prevalent, so we have more flaws.



Source: https://owasp.org/www-project-top-ten/

- Many are due to lack of strong authentication, bad programming, and poor security administration.

THE UNIVERSITY OF SYDNEY