THE UNIVERSITY OF MELBOURNE

Introduction to Cryptography

Olya Ohrimenko

Project 1 is out

- Announcement on LMS
- Spec is available via LMS
- Extra consultation hours

Syllabus

- Operating System concepts
  - Processes
  - Process Scheduling
  - Memory management
- Version Control (Git)
- Security & Cryptography
- Network Services and Development

Security & Cryptography

- Introduction to cryptography
- Secure communication
- Public Key Infrastructure
- System security

Today

Introduction to Cryptography

- Symmetric encryption
- Asymmetric encryption
- Digital signatures

What cryptography is not

It is not bitcoin or cryptocurrency or blockchains

Cryptography: where

My credit card number is XXXX-XXX

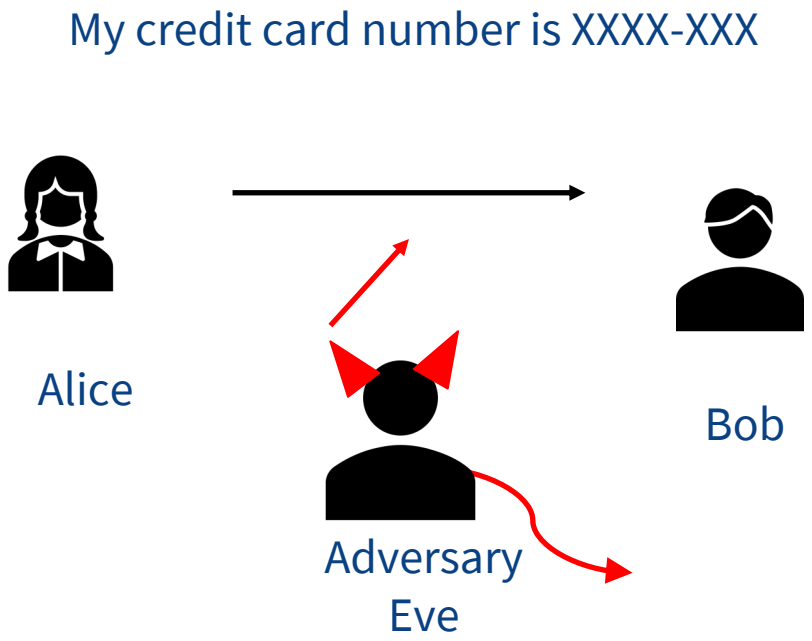
Alice

Bob

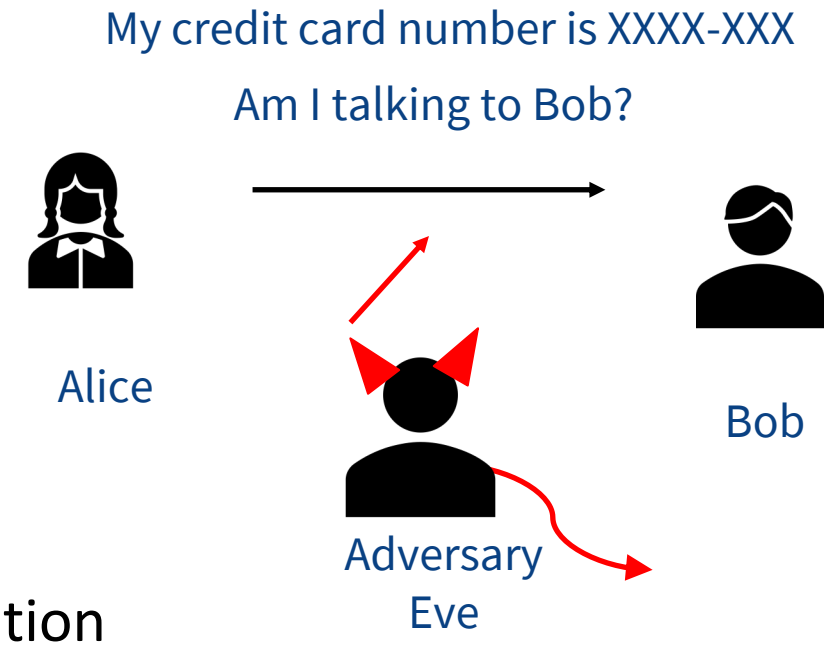
- Secure communication

Cryptography: where

- Secure communication

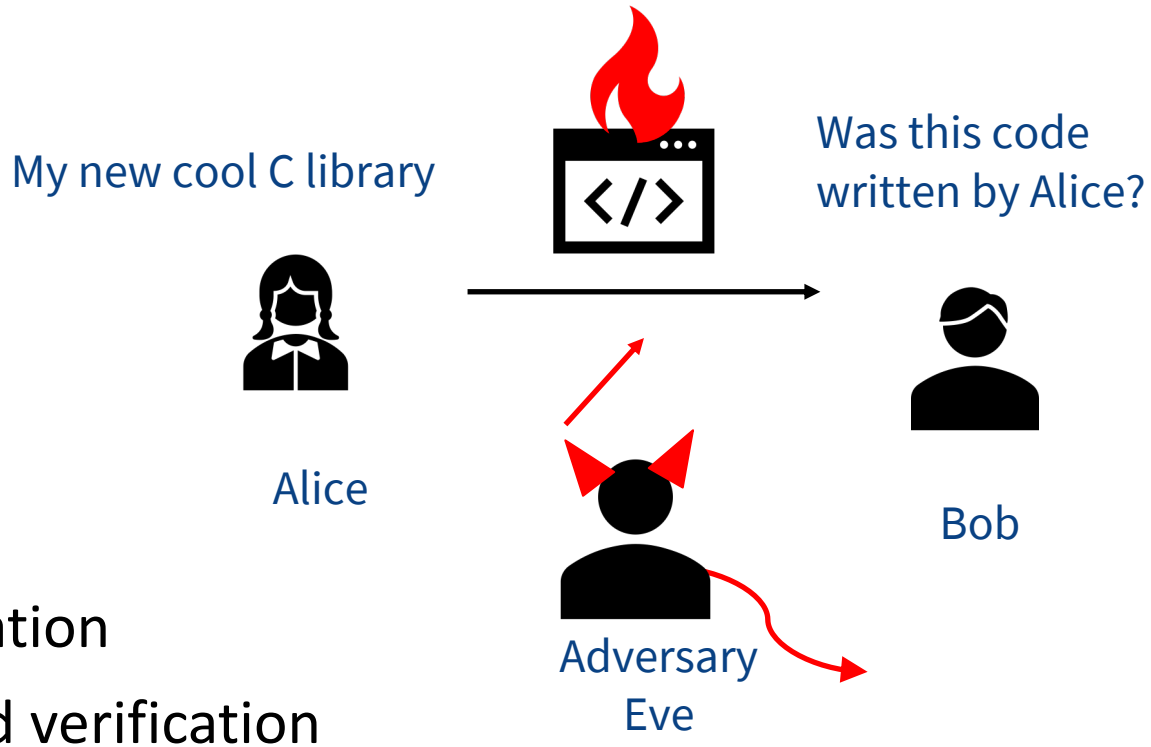


Cryptography: where



- Secure communication
- Authentication and verification

Cryptography: where

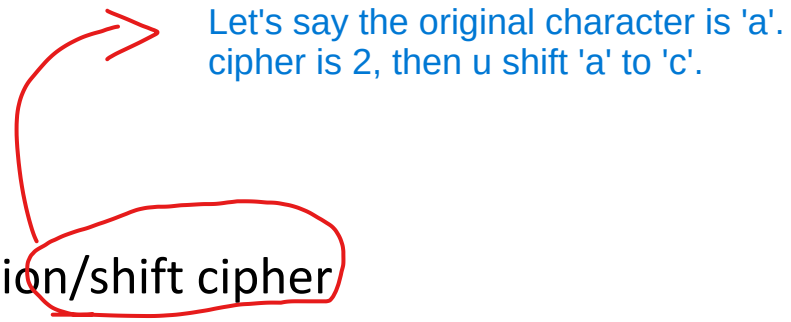


- Secure communication
- Authentication and verification
- Integrity

Encryption/Decryption

- Encryption
  - Hiding data (plaintext) from everyone except those holding the decryption key
  - Output is a cipher text
- Decryption
  - Recovering the original data from the cipher text using the key
  - Output is a plaintext

Encryption/Decryption



- Long and varied history
  - Caesar cipher – simple substitution/shift cipher
  - Enigma
    - Alan Turing (founder of modern computing) led the breaking of it
- Modern cryptography developed significantly since WW2

What is Cryptography?

- Modern cryptography is based on mathematics
  - Problems that are considered to be hard, or are well studied
    - Factorising the product of 2 large primes (RSA)
    - Solving discrete logs (ElGamal)
    - AES – substitution-permutation network
- Information theoretic crypto: one time pad
  - m $\oplus$ x https://www.youtube.com/watch?v=vEbaF1jmbcM
  - where m is a message, x is a secret key, $\oplus$ is XOR

How secure is cryptography

Cryptography is not absolute – there is no perfect security

- Always susceptible to brute force attack
  - Trying every possible key value
- The challenge is to make the brute force attack take so long as to be infeasible to perform within the useful lifetime of the data

Keys in Cryptography

- Symmetric Cryptography
  - Same key is used for encryption and decryption
- Asymmetric Cryptography (Public Key Crypto)
  - Two keys, encrypt with one, decrypt with the other
  - One of the most important developments in modern computer science
- We won't be able to cover all the details
  - Be aware that in cryptography the devil is in the detail

Symmetric Cryptography

- Modern example is AES (Advanced Encryption Standard)
- CipherText := Encrypt(SecretKey, Message)
- Message := Decrypt(SecretKey, CipherText)
- If you want to use it to share a message with someone you have to find some way of securely exchanging the secret key
- Useful for keeping your own data secure
  - Full disk encryption

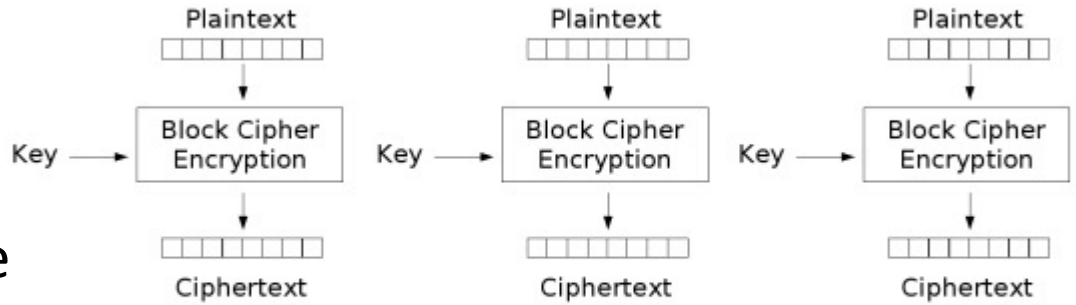
Symmetric Cryptography

IT DOES NOT ONLY ENCRYPT BLOCKS BUT ALSO TIE THOSE BLOCKS.

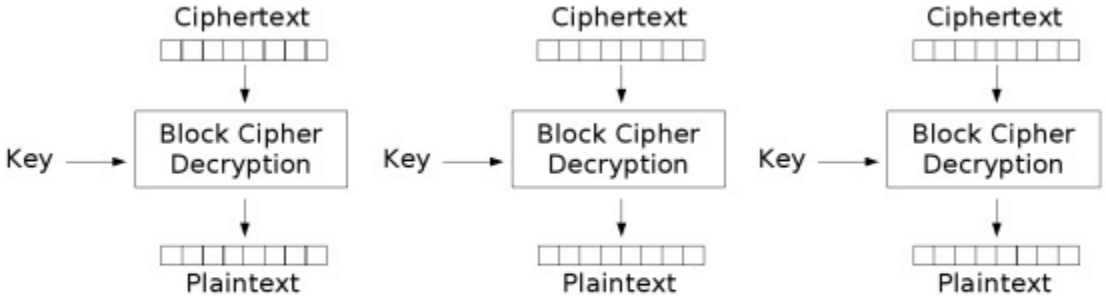
- AES breaks data into blocks and encrypts each block
- AES has different modes of operation
  - ECB, CBC, CTR, etc.
  - Determines how each block is treated, and/or linked
- AES-NI instruction set extension on Intel

ECB – Electronic Codebook

- Simple, parallelisable



Electronic Codebook (ECB) mode encryption



Electronic Codebook (ECB) mode decryption

ECB – Electronic Codebook

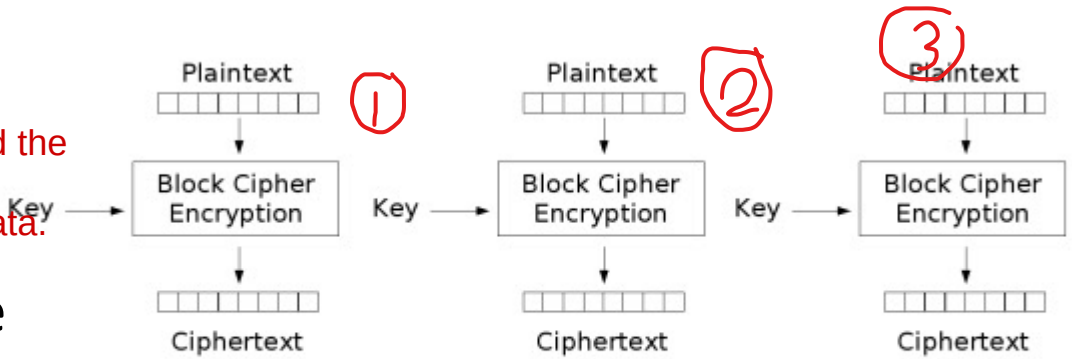
This one doesnt have the XOR PROCESS

THE PROBLEM WITH THIS METHOD IS

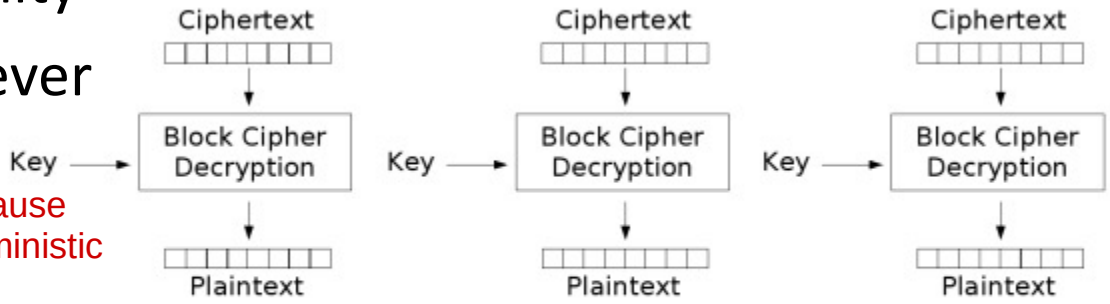
1/ The same plaintext is always encrypted the same way.
2/ A plaintext is divided into 3 blocks of data. Those 3 blocks can be different but their

- Simple, parallelisable
- Does not provide diffusion, may not even provide confidentiality
- Generally should never be used

3/ciphertexts are exactly the same because they r from the same plaintext -> deterministic encrypti



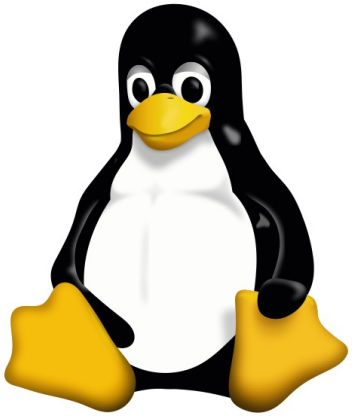
Electronic Codebook (ECB) mode encryption

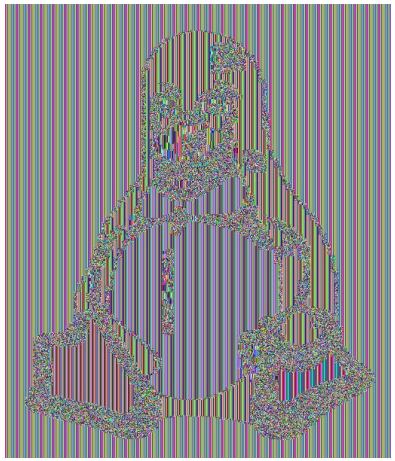


Electronic Codebook (ECB) mode decryption

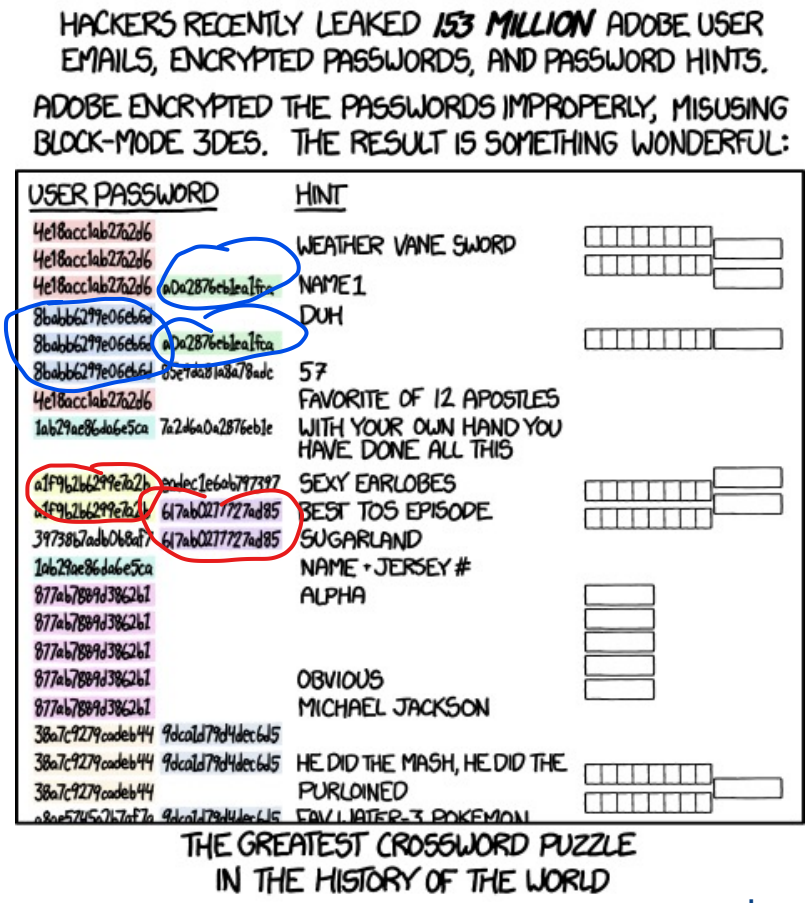
ECB Penguin

- Good example of the problems with ECB
- The problem is that the same plaintext encrypts to the same cipher text
- Repeated patterns will be evident in the output, albeit at a lower fidelity





Symmetric Cryptography



BY ENCRYPTING USED PASSWORDS WITH DETERMINISTIC ENCRYPTION, WE CAN TELL 2 SIMILAR ENCRYPTED PASSWORDS ARE FROM THE SAME USED PASSWORD THOUGH WE DO NOT ACTUALLY KNOW THE USED PASSWORD.

https://xkcd.com/1286/

Probabilistic encryption

ENCRYPT THE SAME PASSWORD MULTIPLE TIMES -> GET DIFFERENT CIPHER TEXTS.

- Turing award 2012 to Shafi Goldwasser and Silvio Micali
- Secure notion of encryption
- Enc(SK, …, “msg”) != Enc(SK, …, “msg”)

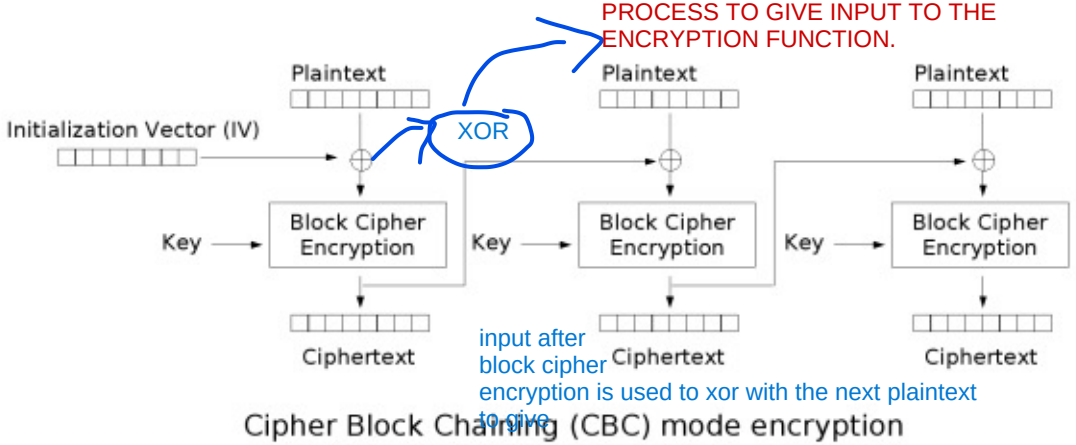
Image source: Wikipedia

CBC – Cipher Block Chaining

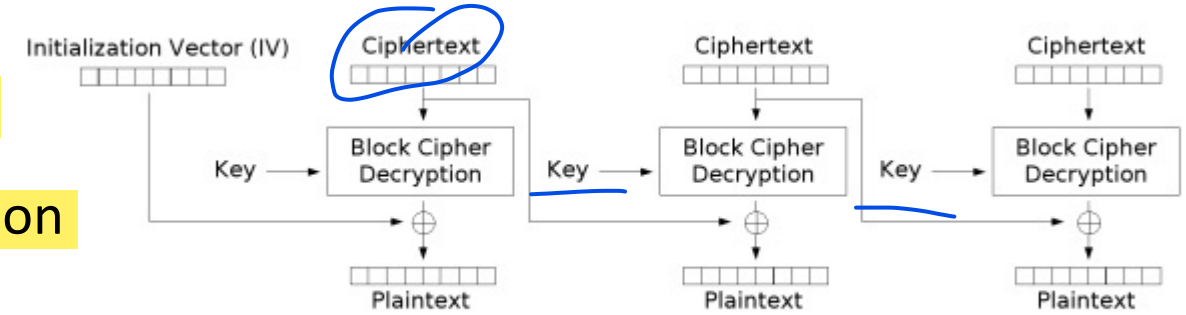
THE MAIN DIFFERENCE BETWEEN THIS METHOD AND THE PREVIOUS METHOD IS THAT THIS ONE HAS A XOR PROCESS TO GIVE INPUT TO THE ENCRYPTION FUNCTION.

- IV – not secret, but must be random and not reused
- Encryption must be done sequentially
- Decryption can be done in parallel
- Loss or corruption of a ciphertext block or IV affects correct decryption of at most [redacted] blocks

DECRYPT THE CIPHER TEXT 1



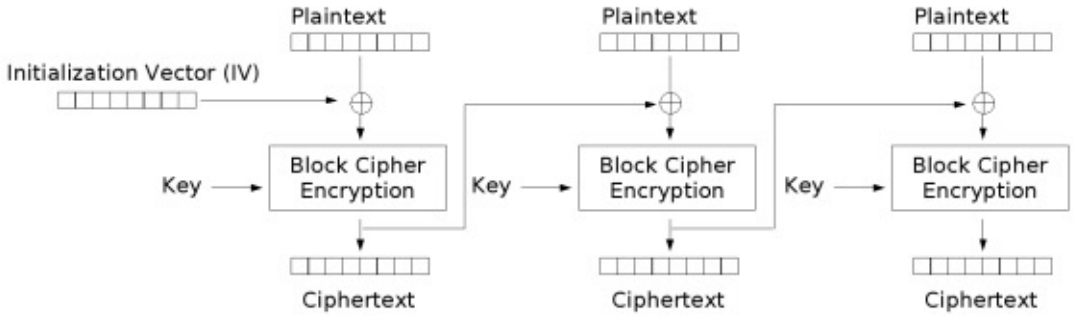
Cipher Block Chaining (CBC) mode encryption



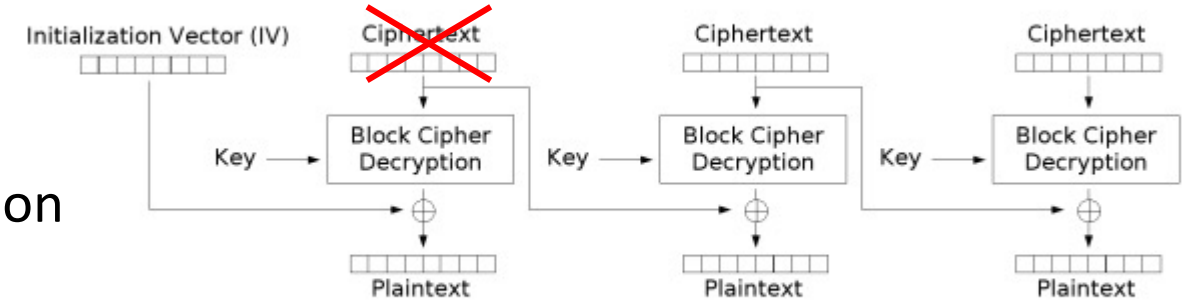
Cipher Block Chaining (CBC) mode decryption

CBC – Cipher Block Chaining

- IV – not secret, but must be random and not reused
- Encryption must be done sequentially
- Decryption can be done in parallel
- Loss or corruption of a ciphertext block or IV affects correct decryption of at most two blocks



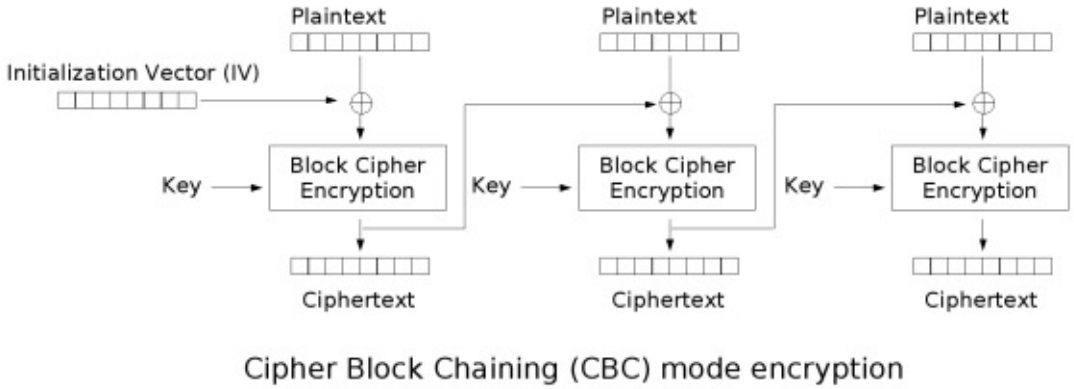
Cipher Block Chaining (CBC) mode encryption



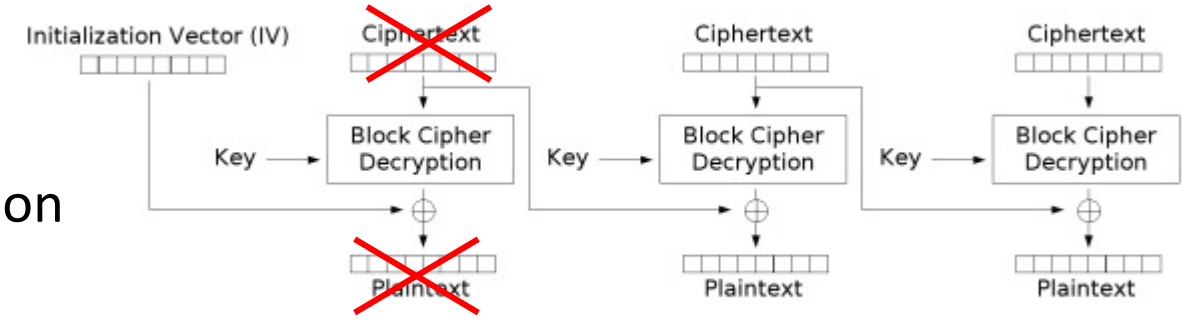
Cipher Block Chaining (CBC) mode decryption

CBC – Cipher Block Chaining

- IV – not secret, but must be random and not reused
- Encryption must be done sequentially
- Decryption can be done in parallel
- Loss or corruption of a ciphertext block or IV affects correct decryption of at most two blocks



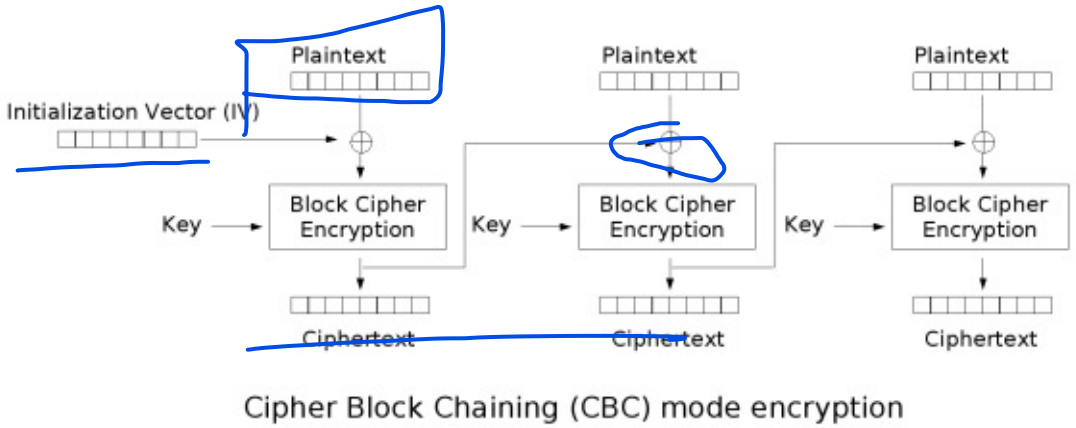
Cipher Block Chaining (CBC) mode encryption



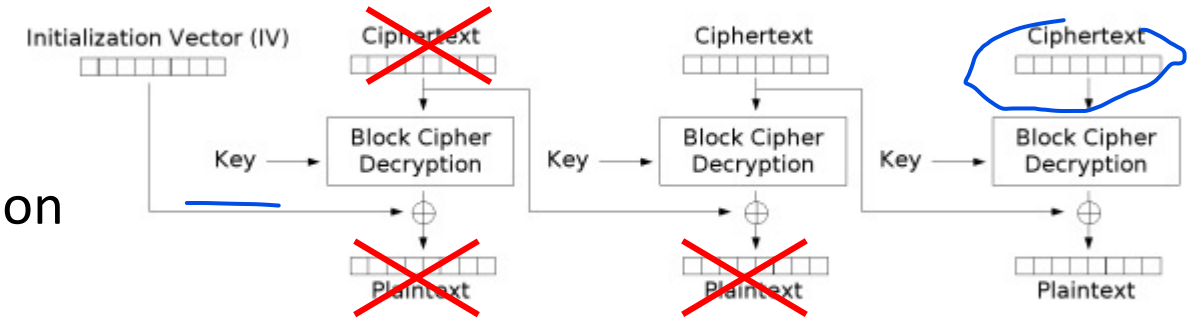
Cipher Block Chaining (CBC) mode decryption

CBC – Cipher Block Chaining

- IV – not secret, but must be random and not reused
- Encryption must be done sequentially
- Decryption can be done in parallel
- Loss or corruption of a ciphertext block or IV affects correct decryption of at most two blocks



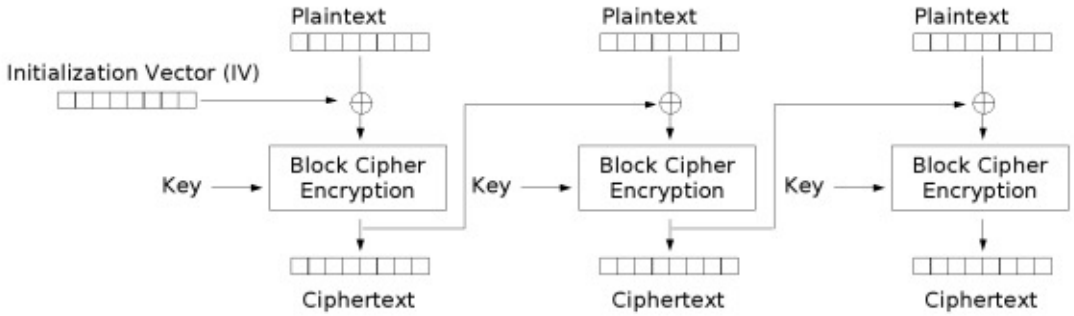
Cipher Block Chaining (CBC) mode encryption



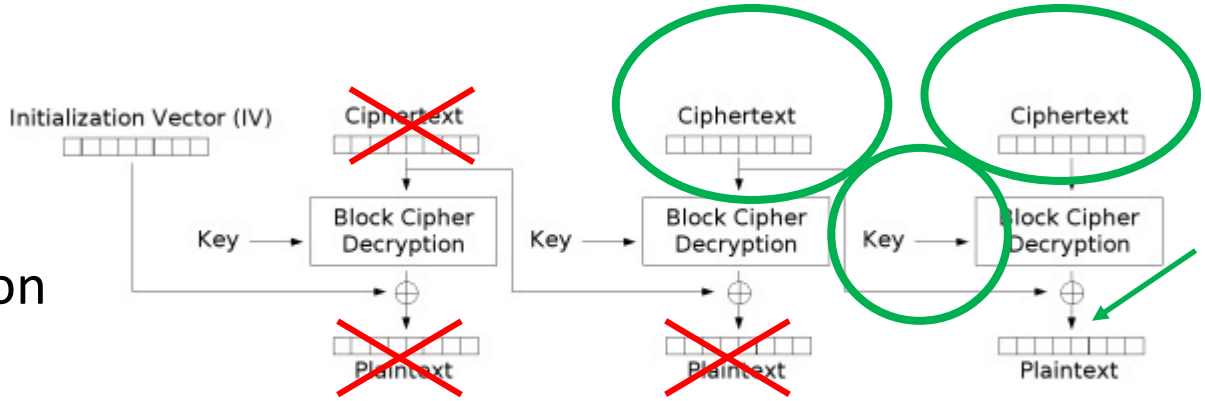
Cipher Block Chaining (CBC) mode decryption

CBC – Cipher Block Chaining

- IV – not secret, but must be random and not reused
- Encryption must be done sequentially
- Decryption can be done in parallel
- Loss or corruption of a ciphertext block or IV affects correct decryption of at most two blocks



Cipher Block Chaining (CBC) mode encryption



Cipher Block Chaining (CBC) mode decryption

Asymmetric Cryptography

- More commonly called public key cryptography
- At the heart of modern security
  - Digital signatures
  - TLS (Transport Layer Security)
  - Secure Messaging
  - End-to-end Encryption (WhatsApp, Signal, etc.)
- Consists of two related keys
  - Public Key – as per the name, can be made public
  - Private/Secret Key – must be kept secret by the owner/user

Asymmetric Cryptography

- Alice generates her key pair (PublicKey, PrivateKey)
- She posts her PublicKey online
- Bob wants to send Alice a secret message m
- Bob runs CipherText := Encrypt(PublicKey, m)
- Bob sends the CipherText to Alice over the internet
- Alice runs m := Decrypt(PrivateKey, CipherText)
  - Alice recovers the secret message
- No need to meet or securely exchange any keys

the problem with using only symmetric crypto is that when Bob encrypt the message that he wants to send to Alice is what if the message is very long. the encryption process is gonna be slow.

Asymmetric & Symmetric

- Asymmetric Cryptography is slower than Symmetric
- It often isn't suitable for encrypting large amounts of data or even multiple blocks
- Often used together with Symmetric Cryptography as a way of exchanging a joint secret key

Asymmetric & Symmetric

Symmetric

- Secret key SK
- Cipher Text := Encrypt(SK, message)
- message := Decrypt(SK, Cipher Text)

Asymmetric

- PublicKey, PrivateKey
- CipherText := Encrypt(PublicKey, message)
- message := Decrypt(PrivateKey, Cipher Text)

Bob knows Alice's PublicKey and has a very very very long message m.

Q: How can Bob use Symmetric and Asymmetric Crypto to send m to Alice?

Key Exchange

- Alice generates her key pair (PublicKey, PrivateKey)
- She posts her PublicKey online
- Bob generates a SK with symmetric encryption
- Bob runs CipherText := Encrypt(PublicKey, SK)
- Bob sends the CipherText to Alice
- Alice runs SK := Decrypt(PrivateKey, Cipher Text)
- Now Alice and Bob share a SK and can exchange Symmetrically encrypted messages
- This is sort of how TLS (HTTPS) works
  - A lot of further detail, but in principle, TLS is a key exchange protocol

INSTEAD of encrypting the actual message, Bob encrypted the secret key generated by symmetric encryption.

ciphertext = Bob encrypt(SK, MESSAGE). Alice decrypt the ciphertext to get the message by using SK.

Asymmetric Cryptography in Action

- RSA is probably the most well known Public Key Cryptography System
  - Other examples are ElGamal, and Elliptic Curve ElGamal

```
openssl genpkey -algorithm RSA -out PrivateKey.pem -pkeyopt rsa_keygen_bits:2048
```

  - Generates a 2048 bit Key Pair

```
openssl rsa -pubout -in PrivateKey.pem -out PublicKey.pem
```

  - Extract PublicKey from key pair file

RSA: Rivest–Shamir–Adleman also Turing award!

Asymmetric Cryptography in Action

- Simple example, would not normally directly encrypt a message with RSA
- Encrypt msg.txt with PublicKey.pub (short message)

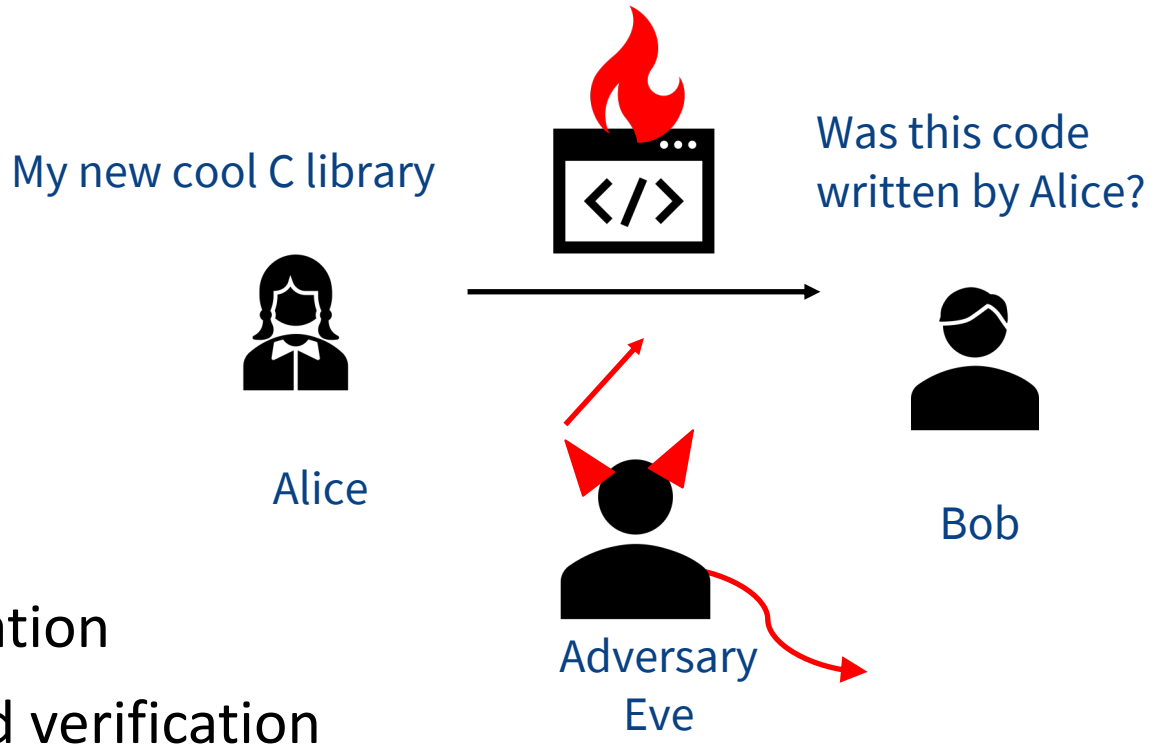
```
openssl rsautl -in msg.txt -out ciphertext.enc -pubin -inkey PublicKey.pub -encrypt
```

- Decrypt ciphertext.enc with PrivateKey returning plaintext.txt

```
openssl rsautl -in ciphertext.enc -out plaintext.txt -inkey PrivateKey.pem –decrypt
```

- Normal approach would be to generate an AES key then encrypt that with the PublicKey, and encrypt the file with the symmetric key

Cryptography: where



- Secure communication
- Authentication and verification
- Integrity

Public Key Signatures

A key holder holds the signed key.

- Provides a link between a key holder and a message/document/file
- Importantly, is considered to provide non-repudiation
  - Cannot deny having signed the document
- Provides integrity (against tampering)

Public Key Signatures

- Generate SignKey (kept private) and VerifKey (Public)
- s := Sign(SignKey, m)
- Anyone can check if verification succeeds:
  - Verify(VerifyKey, s, m) ?

Public Key Signatures

What happens with large documents?

- Use a Cryptographic Hash Function
  - Takes a near arbitrary length input and outputs a fixed length digest
  - SHA256, SHA512, (MD5, SHA1 – now deprecated)
- Sign the Hash Digest – which can be recalculated by the verifier:

  s' := Sign(SignKey, ḣ) where h = Hash(m)

Cryptographic Hashing

- Hash function H is a lossy compression function
  - Collision: H(x)=H(x') for some inputs x≠x'
- H(x) should look "random"
  - Every bit (almost) equally likely to be 0 or 1
- A cryptographic hash function must have certain properties:
  - Collision resistance: hard to find x≠x' such that h(x)=h(x')
- Hashing vs. encryption
- Applications: password storage(?), software integrity

Cryptographic Hashing

- Two hashes, small difference, big change

```
echo 123456 | openssl sha256
(stdin)=
e150a1ec81e8e93e1eae2c3a77e66ec6dbd6a3b460f89c1d08aecf422ee401a0
echo 123457 | openssl sha256
(stdin)=
4547694d80b0c7675bca81b8e9ea8efdeda10c92b910c19a83c08e2f06bf5cc8
```

- Small file hash (13 bytes)

```
openssl sha256 plaintext.txt SHA256
(plaintext.txt)=
03ba204e50d126e4674c005e04d82e84c21366780af1f43bd54a37816b6ab340
```

- Large file hash (1.6 GB)

```
openssl sha256 ubuntu-16.04.3-desktop-amd64.iso SHA256
(ubuntu-16.04.3-desktop-amd64.iso)=
1384ac8f2c2a6479ba2a9cbe90a585618834560c477a699a4a7ebe7b5345ddc1
```

Public Key Signatures

- Alice wants to sign a document, calls:
  - Sign(PrivateKey, Hash(document)) -> Digital Signature

```
openssl dgst -sha256 -sign PrivateKey.pem -out signature.txt plaintext.txt
```

- Bob wants to verify Alice signed the document, calls:
  - Verify(PublicKey, Hash(document)) -> True or False

```
openssl dgst -sha256 -verify PublicKey.pem -signature signature.txt plaintext.txt
Verified OK
```

  - If we modify the document by a single character

```
openssl dgst -sha256 -verify PublicKey.pem -signature signature.txt modified.txt
Verification Failure
```

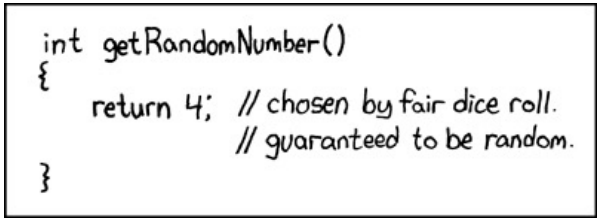
Key Sizes

- Recall our security is based on a computationally hard problem, taking an infeasibly long time to solve using brute force
- As such, our security parameter, which determines the degree of security we achieve, is the key size
  - Normally expressed in bits
- Different key lengths are required for symmetric and asymmetric
  - Reflects the different mathematically hard problems they are based on
  - Asymmetric is generally longer than Symmetric, current minimums:
    - RSA – 2048 bits (3072*), AES – 128 bits
    - Previously much shorter – 512 bits for RSA in the 90's, 56 bits for DES

*https://www.cyber.gov.au/

Randomness

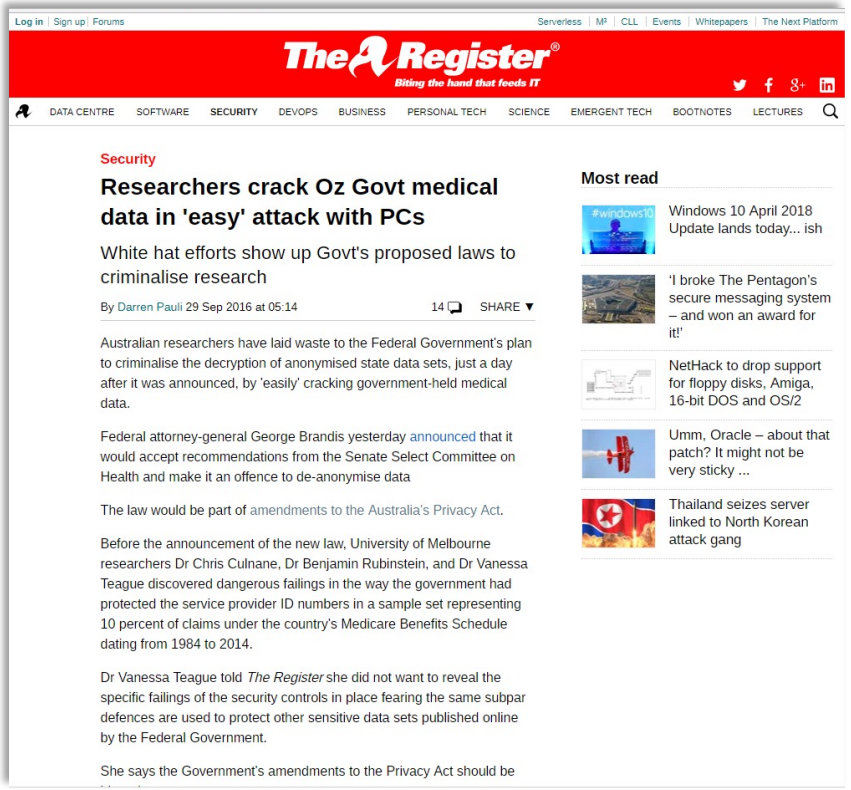
- Cryptography is dependent on randomness
  - Key Generation
  - IV generation for ciphers
  - Padding
- Pseudorandom generators
- Randomness must never be reused or discoverable
- Generating good randomness is hard
  - Must be unpredictable
  - Dependent on good OS implementations



https://xkcd.com/221/

Do not roll your own cryptography

- Just because something looks random, doesn't mean it is securely encrypted
- Even the smallest mistake can weaken the entire approach
- When we talk about large numbers in cryptography we mean very large numbers



Log in | Sign up | Forums

Serverless | MP | CLL | Events | Whitepapers | The Next Platform

The Register®
Biting the hand that feeds IT

DATA CENTRE | SOFTWARE | SECURITY | DEVOPS | BUSINESS | PERSONAL TECH | SCIENCE | EMERGENT TECH | BOOTNOTES | LECTURES

Security

Researchers crack Oz Govt medical data in 'easy' attack with PCs

White hat efforts show up Govt's proposed laws to criminalise research

By Darren Pauli 29 Sep 2016 at 05:14 14 SHARE ▼

Australian researchers have laid waste to the Federal Government's plan to criminalise the decryption of anonymised state data sets, just a day after it was announced, by 'easily' cracking government-held medical data.

Federal attorney-general George Brandis yesterday announced that it would accept recommendations from the Senate Select Committee on Health and make it an offence to de-anonymise data

The law would be part of amendments to the Australia's Privacy Act.

Before the announcement of the new law, University of Melbourne researchers Dr Chris Culnane, Dr Benjamin Rubinstein, and Dr Vanessa Teague discovered dangerous failings in the way the government had protected the service provider ID numbers in a sample set representing 10 percent of claims under the country's Medicare Benefits Schedule dating from 1984 to 2014.

Dr Vanessa Teague told *The Register* she did not want to reveal the specific failings of the security controls in place fearing the same subpar defences are used to protect other sensitive data sets published online by the Federal Government.

She says the Government's amendments to the Privacy Act should be

Most read

Windows 10 April 2018 Update lands today... ish

'I broke The Pentagon's secure messaging system – and won an award for it!'

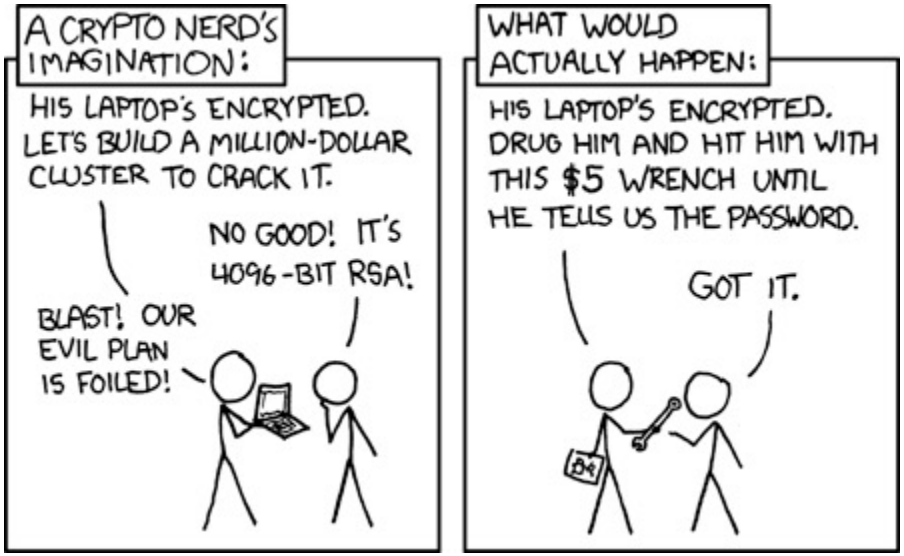
NetHack to drop support for floppy disks, Amiga, 16-bit DOS and OS/2

Umm, Oracle – about that patch? It might not be very sticky ...

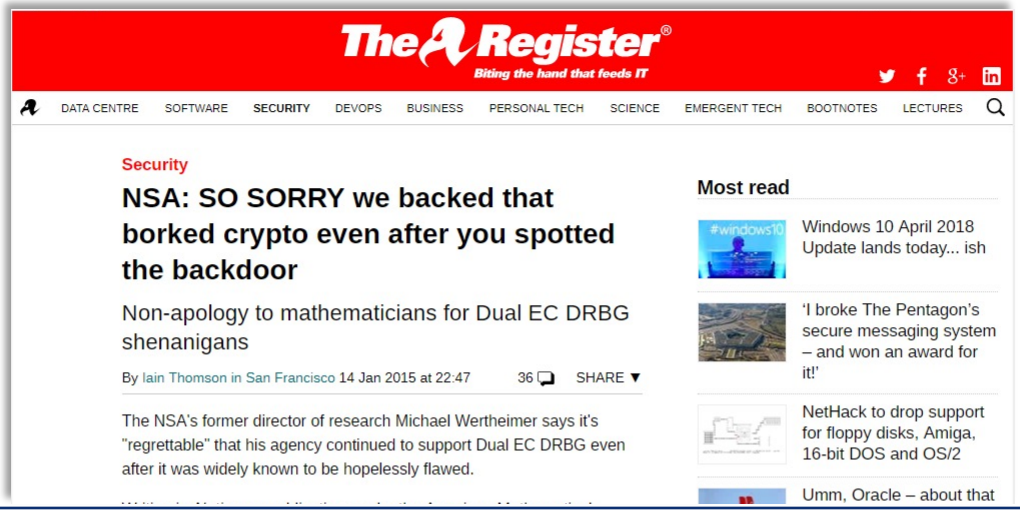
Thailand seizes server linked to North Korean attack gang

And finally...

- Compromising randomness generation is a known attack
  - NSA Dual EC DRBG
  - Juniper Networks



https://xkcd.com/538/



Summary

- Symmetric vs asymmetric cryptography
- Encryption
- Signatures
- Hashing

Acknowledgement

- The slides were prepared by Olya Ohrimenko based on material developed previously by: Chris Culnane, Michael Kirley, Zoltan Somogyi, Rao Kotagiri, James Bailey and Chris Leckie.
- Some of the images included in the notes were supplied as part of the teaching resources accompanying the text books listed in lecture 1.
  - (And also) Wikimedia Commons
- Reference: KR 8, 8.1, 8.2 till RSA (unless curious to learn more about RSA), 8.3, 8.3.1, 8.3.3 till page 644