Distributed Systems — Security

Allan Clark

School of Informatics
University of Edinburgh

http://www.inf.ed.ac.uk/teaching/courses/ds
Autumn Term 2012

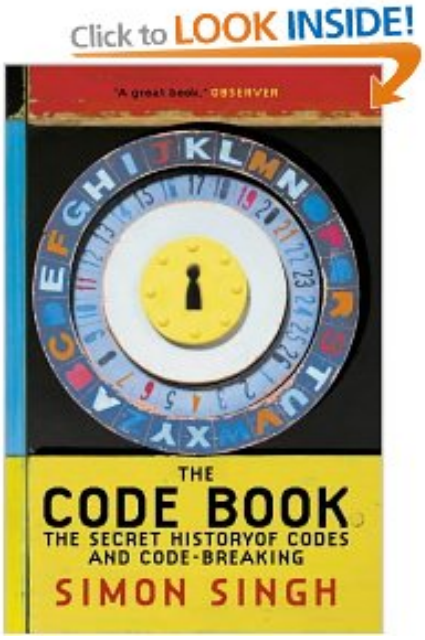
Security

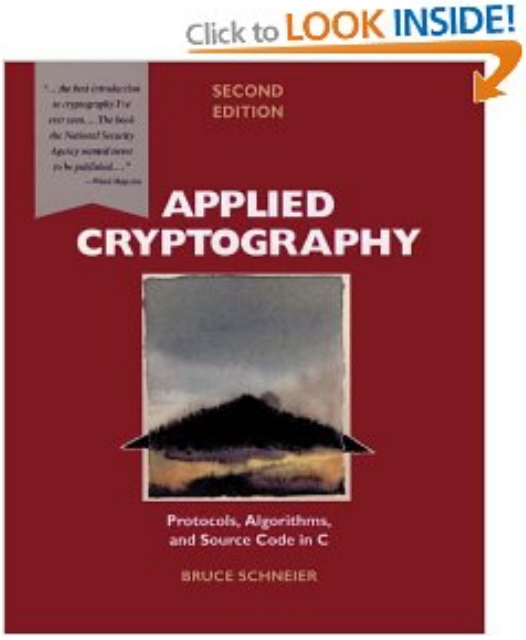
Overview

- In this part of the course we will look at security in distributed systems
- Cryptography will provide the basis of secrecy and integrity
  - That is, making sure that no unauthorised entity may read any particular message
  - No unintended message is delivered, including a duplication of an intended message
- We will examine private-key techniques as well as public-key techniques and digital signatures
- We will look at cryptographic algorithms

Security

Books

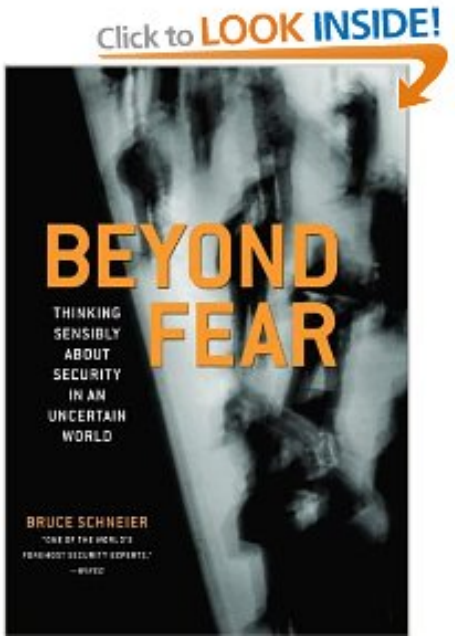




Security

Books

- We will focus on threats to distributed systems caused by the inavoidable exposure of their communication channels
- The largest threat is generally human error
- Bruch Schneier also has a newsletter each month called "cryptogram" which talks about many security related topics including cryptography and physical/human related policies



Security

Cryptography

- Although computer security and computer cryptography are separate subjects, digital cryptography provides the basis for most of the mechanisms that we use in computer security
- It is only in recent years (the 1990s) that cryptographic techniques have been wrestled from the domain of the military into the domain of public knowledge and use
- When Bruce Schneier first published his book "*Applied Cryptography*" in 1994 the legal status of including cryptographic algorithms and techniques was in doubt.

Security

Pre-1999 US Munitions Control

- RSA crypto-algorithms, were, until 1999, classified by the US State Department as munitions
- Meaning they were classified in the same category as: chemical and biological weapons, tanks, heavy artillery, and military aircraft
- Additionally this meant that it was illegal to export such cryptographic algorithms, with penalties including $1m fines and long prison sentences
- This was obvious buffoonery:
  - It is impossible to enforce
  - The technology is widely available throughout the world
  - Algorithms published in international journals
  - Some cryptographic algorithms were developed outside the US

Security

Pre-1999 US Munitions Control

- Popular email programs such as *Netscape Communicator* had to have separate downloads for US based downloaders and external downloaders
- When it went open-source and became *mozilla* this was more nonsense since very quickly the external versions were patched to include full 160-bit encryption
- People took to methods of highlighting how ridiculous such an export ban was, one such effort demonstrated that RSA crypto algorithms can be written in a fairly short amount of Perl code

```
#!/bin/perl -sp0777i<X+d*lMLa^*lN%0]dsXx++lMlN/dsM0<j]dsj
$/=unpack('H*',$_);$_=`echo 16dio\U$k"SK$/SM$n\EsN0p[lN*1
lK[d2%Sa2/d0$^Ixp"|dc`;s/\W//g;$_=pack('H*',/((..)*)$/)
```

Security

Pre-1999 US Munitions Control

- So to highlight how ludicrous it was people started attaching it to emails
- Technically if said emails were sent outwith the US such people could have been prosecuted

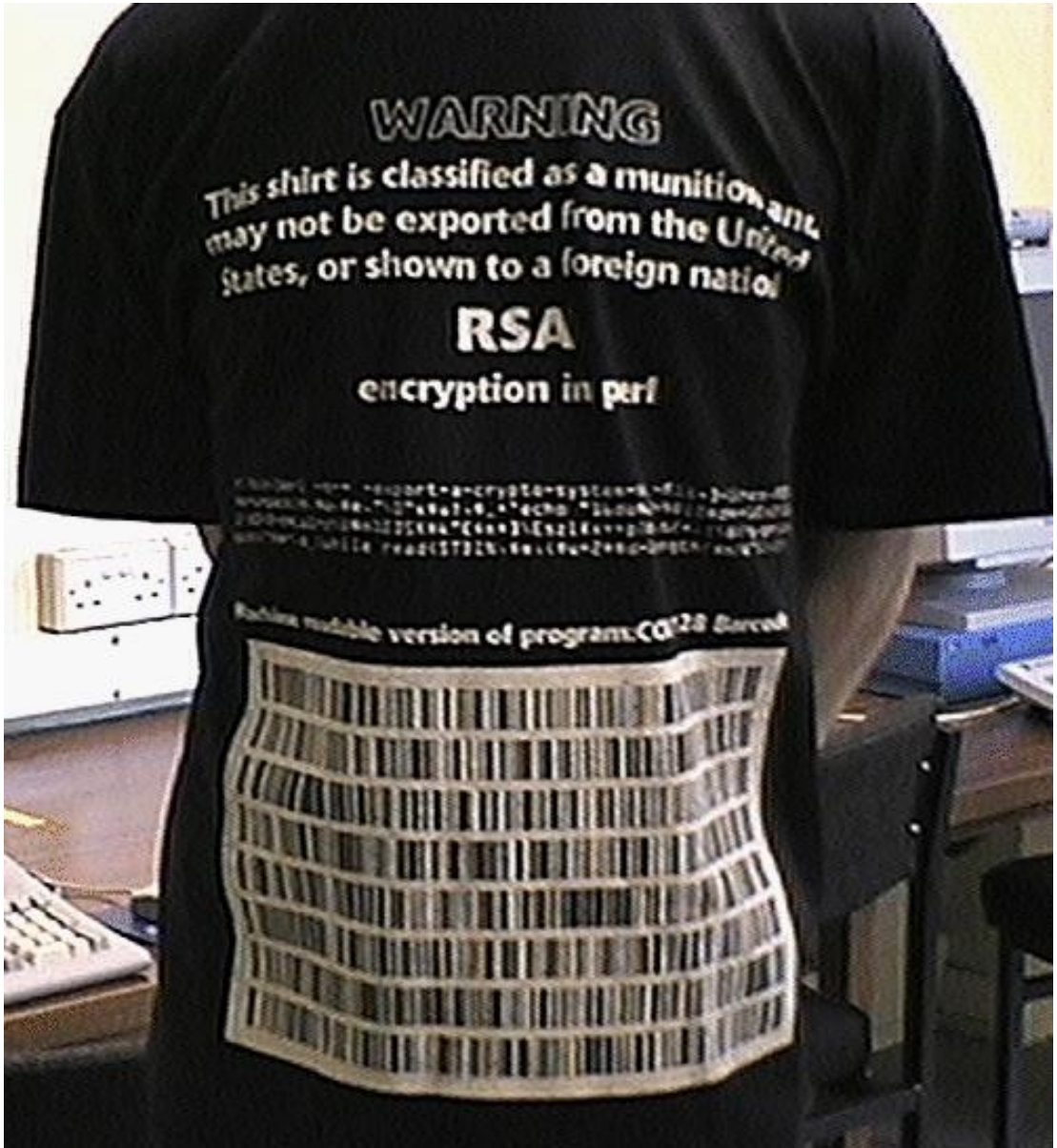
```
--

The following is classified as munitions by
 the US state department:

#!/bin/perl -sp0777i<X+d*lMLa^*lN%0]dsXx++lMlN/dsM0<j]dsj
$/=unpack('H*',$_);$_=`echo 16dio\U$k"SK$/SM$n\EsN0p[lN*1
lK[d2%Sa2/d0$^Ixp"|dc`;s/\W//g;$_=pack('H*',/((..)*)$/)
```

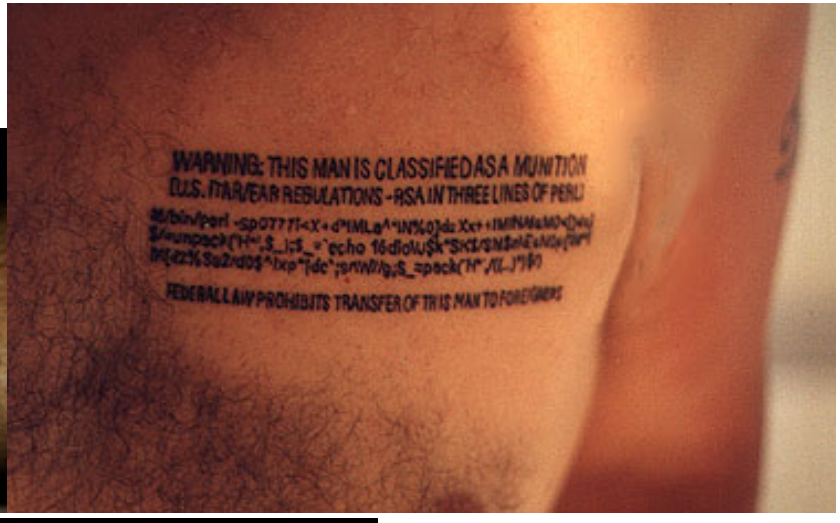
Security

T-Shirt



Security

Tattoos



Security Model

We will assume

- Wherever you are in the world you have access to cryptographic protocols and algorithms
- There are a set of nodes which share resources
  - Resources may be physical or data/programming objects
- Communication is via message passing only, and hence access to shared resources occurs via message passing
- The nodes are connected via a network which may be accessed by any enemy
- An enemy may copy or read any message transmitted through the network
- They may also inject arbitrary messages, to any destination purporting to come from any source

Security

Policies and Mechanisms

- There is a distinction between a security policy and a security mechanism
- Security policies are independent of the security mechanisms used with that policy
- A system cannot be secured using only security mechanisms
- For example, the door to your accommodation is likely secured using a lock and key, that is the security mechanism
- But it is near useless without the accompanying policy:
  - The last person to leave the building should lock the door

Security

Threats and Attacks

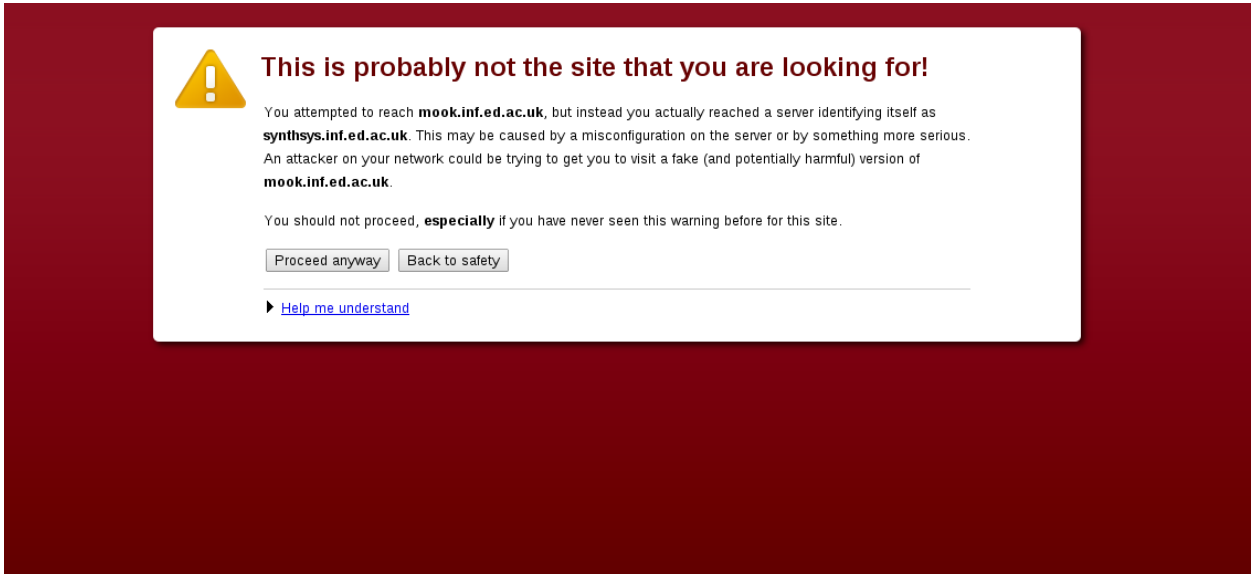
- For most types of network, and certainly wireless networks, it is generally obvious that an attacker wishing to obtain private information can simply listen in on all messages
- Doing so means that it is relatively simple to construct a computer that would simply log all messages between communicating computers
- Depending on the application simply knowing the contents of some messages may be enough, otherwise the attacker may need information about the distributed algorithm in question in order to construct information from the data in the messages that were recorded

Security

Threats and Attacks

- A slightly more elaborate attack is to construct a server in between the client and the intended server
- If the client does not authenticate the server, then it may send private information to what it believes to be the intended server
- Often the fake server will then log the information sent to it, but then also forward it on the real server in question
- Thus the attack is non-trivial to detect.
- This is a common technique for obtaining web-passwords

Security



- Third party "*Certificate Authorities*" issue digital certificates containing encryption keys to verify the identity of secure websites

Security

Threats and Attacks

- Threats and attacks fall into three broad categories:
  1. Leakage
     - The acquisition of data by unauthorised entities
  2. Tampering
     - The alteration of data by an unauthorised entity
  3. Vandalism
     - Distruption to the service in question without gain to the perpetrators

Security

Threats and Attacks

- We can further distinguish attacks in a distributed system by the way in which communication channels are misused:
  1. *Eavesdropping*
     - Obtaining copies of messages without authority
  2. *Masquerading*
     - Sending or receiving messages using the identity of another process/entity without their authority
  3. *Message Tampering*
     - Intercepting messages and altering them before forwarding them on to their intended recipient
  4. *Replaying*
     - Storing intercepted messages and sending them at a later date. This attack can be effective even when used against authenticated and encrypted messages (think of the two generals problem)
  5. *Denial of Service*
     - Flooding a service with requests such that it cannot handle legitimate requests

Security

Information Existence

- Regardless of how strong your encryption may be, the detection of a message transmitted between two processes may leak information
- The mere existence of such a message may be the source of information.
- For example a flood of messages to a dealer of a particular set of stocks may indicate a high-level of trading for a particular stock
- One possible defence is to regularly send nonsense/ignorable messages

Security

Trade-offs

- Ultimately all security measures involve trade-offs
- A cost is incurred in terms of computational work and network usage for use of cryptography and other protocols
- Where a security measure is not correctly specified it may limit the availability of the service for legitimate users/uses
- These costs must be stacked up against the threat or cost of failure to maintain security
- Generally we wish to avoid disaster and minimise mishaps

Security

Assume the worst

- Interfaces are exposed distributed systems are designed such that processes offer a set of services, or an interface. These interfaces must be open to allow for new clients. Attackers therefore are able to send an arbitrary message to any interface
- Networks are insecure An attacker can send a message and falisfy the origin address so as to masquerade as another user. Host addresses may be spoofed so that an attacker may receive a message intended for another
- Algorithms and program code is available to attackers Messages sent may be intercepted but that may not be useful since to make sense of the message an attacker may need to know the purpose/protocol within which the message is sent. Assume that that may be the case

Security

Assume the worst

- Attackers may have access to large resources Do not therefore rely on the fact that you may compute something faster than an attacker, or that an attacker has a limited timeframe in which their attack may be valid/dangerous/worthwhile
- Assume all code may have flaws the part of your software responsible for security must be trusted. Often called the *trusted computing base*. It should be minimised, for example application programmers should not be trusted to protect data from their users

Security

Cryptography

- Modern Cryptography relies on the use of algorithms which distort a message and reverse that distortion using a secrets called *keys*
- A simple substitution cyper is an example of this:
- In this case the key is the mapping of characters:
  - $a \mapsto f, b \mapsto x, c \mapsto j, \ldots$
- Today's encryption techniques are believed to have the property that the decryption key cannot be feasibly guessed using the cypertext (the encrypted message)

Security

Cryptography

- There are two main algorithms in use:
  1. shared secret keys
     - both parties must share knowledge of the secret key and it must not be shared with any other party
  2. public/private key pairs
     - The sender uses the receiver's *public* key to encrypt the message.
     - The encryption cannot be reversed by the *public* key and can only be reversed by the receiver's *private* key
     - The sender needs to know the receiver's *public* key but need not know the receiver's *private* key
     - Anyone may know the receiver's *public* key but the *private* key must be known only to the receiver
- Both kinds of algorithms are very useful and widely used
- public/private key algorithms require 100/1000 times more processing power
- The lack of need for initial secure transfer of the private key often outweighs the disadvantage

Security

Some Notation and Characters

- Alice and Bob are participants in security protocols
  - Alice has the secret key K_A and Bob the secret key K_B
  - They have a shared secret key K_{AB}
- Alice has a private key K_{Apriv} and a public key K_{Apub}
- $\{M\}_K$ is a message encrypted with key K
- $[M]_K$ is a message signed with key K
- Carol and Dave are extra participants for 3,4 party protocols
- Eve is an eavesdropper
- Mallory is a malicious attacker
- Sara is a server

Security

Scenario 1. Secure communication

- Cryptography can be used to enable secure communication
- In this instance each message is encrypted and can only be decrypted with the correct secret key
- So long as that secret key is not compromised then secrecy can be maintained
- Integrity is generally maintained using some redundant information within the encrypted message, such as a checksum

Security

Scenario 1. Secure communication

- Alice wishes to send some secret information to Bob
- If they share the secret key K_{AB} then:
- Alice uses the key and an agreed encryption algorithm $E(K_{AB}, M)$ to encrypt and send any number of messages $\{M_i\}_{K_{AB}}$
- Bob decrypts the messages using the corresponding decryption algorithm $D(K_{AB}, M)$
- Two problems:
  1. How can Alice initiate this communication by sending the secret key K_{AB} to Bob securely?
  2. How does Bob know that a message $\{M_i\}$ isn't a copy of an earlier encrypted message sent by Alice but intercepted by Mallory?

Security

Scenario 2. Authentication

- Cryptography can be used to authenticate communication between a pair of participants
- If there is a shared secret key known only to two parties, then a successful decryption of a received message requires that the message was originally encrypted using the appropriate key
- If only one (other) party knows of that secret key then we can deduce from whom the message originated

Scenario 2. Authentication

- Alice wishes to communicate with Bob
- Sara is a securely managed authentication server
- Sara stores a secret key for each user, each user knows (or can generate from a password) their own secret key.
- Sara may generate a ticket which consists of a new shared key together with the identity of the participant to whom the ticket is issued

Scenario 2. Authentication

- Steps to secure communication:
  1. Alice sends a request to Sara stating who she is and requesting a ticket for secure communication with Bob.
  2. Sara creates a new secret key K_{AB} to be shared between Alice and Bob. Sara encrypts the ticket using Bob's secret key and sends that together with the secret key all encrypted with Alice's secret key $\{(\{ticket\}_{K_B}, K_{AB})\}_{K_A}$
  3. Alice decrypts this message and obtains the shared secret key and a message containing the ticket encrypted using Bob's secret key. Alice cannot decrypt this ticket message
  4. Alice sends the ticket together with her identity and a request for shared communication to Bob
  5. Bob decrypts the ticket: $\{(K_{AB}, Alice)\}_{K_B}$, confirms that the ticket was issued to the sender (Alice). Alice and Bob can then communicate securely using the (now) shared secret key K_{AB}. Generally the key is used for a limited amount of time before a new one is requested from Sara.

Security

Scenario 2. Authentication

- This is a simplified version of Needham and Schroeder algorithm which is used in Kerberos system (developed at MIT and used here)
- The simplified version does not protect against a replay attack, where old authentication messages are replayed
- It is used within organisations since the individual private keys, K_A, K_B etc, must be shared between the authentication server and the participants in some secure way
- It is therefore inappropriate for use with wide area applications such as eCommerce
- An important breakthrough was the realisation that the user's password need not be sent through the network each time authentication is required. Instead "challenges" are used
- When the server sends Alice the ticket and new shared private key it encrypts it with Alice's own private key. An attacker pretending to be Alice would be defeated at this point

Security

Scenario 3. Authenticated Communcation with Public Keys

- Assuming that Bob has generated his own public/private key pair K_{Bpub}, K_{Bpriv} then Alice and Bob can securely set up a shared private key K_{AB}
- We also assume that there is some public-key certificate system such that Alice can obtain Bob's public key in a way that she is confident that it is indeed Bob's public key
  1. Alice obtain's Bob's public key K_{Bpub}
  2. Alice creates a new shared key K_{AB} and encrypts it using K_{Bpub} using a public-key algorithm. This she sends to Bob $\{K_{AB}\}_{K_{Bpub}}$
  3. Bob decrypts this using the appropriate private key to obtain the shared private key K_{AB}. Shared communication can now take place

Security

Scenario 3. Authenticated Communcation with Public Keys

- This is a hybrid cryptographic protocol and is widely used as it exploits useful features of both public-key and secret-key encryption algorithms
- The slower public-key algorithm is used to set up the speedier secret-key communication
- Problem:
  - The distribution of public keys. Mallory may intercept Alice's initial request to obtain Bob's public key and simply send Alice their own public key.
  - Mallory then intercepts the sending of the shared key which they copy and then re-encrypt using Bob's real public key and forward it to Bob.
  - Mallory can then intercept all subsequent messages since they have the shared secret key. They may need to in order to forward the messages on to Bob and Alice depending on the delivery mechanism.

Security

Digital Signatures

- Cryptography can be used to implement digital signatures
- Alice can encrypt a message using Bob's public key such that only Bob can decrypt the message
- Alice can also encrypt the message using her own secret key
- Anyone can decrypt the message so long as they know Alice's public key
- Provided we can be sure that the public key in question really is that of Alice's we now know that the message must have originated from Alice, since only Alice knows Alice's secret key
- Rather than encrypt the entire message Alice can compute a digest of the message, where a digest is similar to a checksum except that two distinct messages are very unlikely to have the same digest value
- This digest is encrypted and attached to the message, the receiver can then check that the unencrypted digest matches the (receiver computed) digest of the contents of the message

Security

Scenario 4. Digital Signatures

- Alice wishes to sign a document M so that any subsequent receiver can be sure that it originated from Alice
  1. Alice computes a fixed length digest of the document $Digest(M)$
  2. Alice encrypts the digest with her private key and attaches the result to the message $M, \{Digest(M)\}_{K_{Apriv}}$
  3. Alice makes the document with signature available
  4. Bob obtains the signed document, extracts M and computes $d = Digest(M)$
  5. Bob decrypts $\{Digest(M)\}_{K_{Apriv}}$ using K_{Apub} and compares the result to d, if they match the signature is valid.

Security

Scenario 4. Digital Signatures

- We have three requirements of digital signatures
  1. Authentic It convinces the recipient that the signer deliberately signed the document and it has not been altered by anyone else
  2. Unforgeable It provides proof that no one else deliberately signed the document. In particular the signature cannot be copied and placed on another document
  3. Non-repudiable The signer cannot credibly deny that the document was signed by them
- Note that encryption of the entire document, or its digest, gives good evidence for the signature as unforgeable
- Non-repudiable is the most difficult to achieve for digital signatures. A signer may simply deliberate disclose their secret key to others and then claim that anyone could have signed it
- This can be solved through engineering but is generally solved through social contract "If you give away your secret key you are liable"

Security

5. Certificates

- Suppose Alice would like to shop with Carol
- Carol would like to be sure that Alice has some form of bank account
- Alice has a bank account at Bob's bank
- Bob's bank provides Alice with a *certificate* stating that Alice does indeed have an account with Bob.
- Such a certificate is digitally signed with Bob's private key K_{Bpriv} and can be checked using Bob's public key K_{Bpub}

Security

5. Certificates

- Now suppose Alice wished to carry out an attack such that she convinced Carol that someone else's account was owned by herself
- This is quite simple, Alice only requires to generate a new public-private key pair $K_{BprivFake}, K_{BpubFake}$
- She then creates a certificate falsely claiming that Alice is the owner of some account and signs it using $K_{BprivFake}$
- If she can convince Carol that $K_{BpubFake}$ is the true public key of Bob's bank, then this attack should work no problem

Security

5. Certificates

- The solution is for Carol to require a certificate from a trusted fourth party, Dave from the Bankers' Federation, whose role it is to certify the public keys of banks
- Dave issues a public-key certificate for Bob's public key K_{Bpub}. This is signed using Dave's private key K_{Dpriv} and can be verified using Dave's public key K_{Dpub}
- Of course now we have a recursive problem, since now we need to authenticate that K_{Dpub} is the legitimate public key of Dave from the Bankers' federation.
- We break the recursion by insisting that at some point Carol must trust one person, say Dave, and to do so may require to meet them in person.
- Note that Carol only has to trust Dave in order to verify bank account certificates from a variety of banks

Security

5. Certificates

- To make certificates useful, we require:
  1. A standard format such that certificate issuers and users can construct and interpret them successfully.
  2. Agreement on the way in which chains of certificates are constructed and in particular the notion of a trusted authority
- In addition, we may wish to revoke a certificate, for example if someone closes their account
- This is problematic since once the certificate is given it can be copied and stored etc
- The usual solution is for the certificate to have an expiration date, meaning that the holder of the certificate must periodically renew it (in the say way that one renews a passport)

Security

Cryptographic Algorithms

- Until now we have just assumed there is some method of encrypting the plaintext into the corresponding cyphertext using a particular key
- Additionally that there is some inverse operation to decrypt the cyphertext back into the original plaintext, using the same or corresponding decryption key
- The encryption depends on two things, the method E and the key K
- A message M has an encrypted version $\{M\}_K$ if:
  - $\{M\}_K = E(K, M)$
- The mathematically minded can think of an encryption algorithm as describing a (large) family of encryption functions from which one is selected by any given key
- Decyption of course gives the original message when used with the correct key
  - $M = D(K', \{M\}_K)$

Security

Symmetric Algorithms

- Shared secret key, or symmetric algorithms use the same key for decryption as for encryption, such that:
- $M = D(K, E(K, M))$ or $M = D(K, \{M\}_K)$
- It should be the case that the inverse function $M = E^{-1}(\{M\}_K)$ is so hard to compute as to be infeasible
- However both $E(K, M)$ and $D(K, \{M\}_K)$ should be relatively easy to compute
- Such functions are known as *one-way* functions

Security

Defence — symmetric algorithms

- Whilst a strong *one-way* function defends against an attack which attempts to discover M given $\{M\}_K$
- It does not necessarily defend against an attack which seeks to discover K given M and $\{M\}_K$ (and crucially E)
- This has been an important attack and was used heavily during World War II to break the Nazi *enigma* encryption scheme
- The simplest and highly effective attack is a *brute-force* attack in which all keys are attempted, computing $E(K, M)$ to see if it matches $\{M\}_K$
- The number of possible keys depends on the length of K, if it has N bits then there are 2^N possibilities (though you need only try 2^{N-1} on average.

Security

Block Ciphers

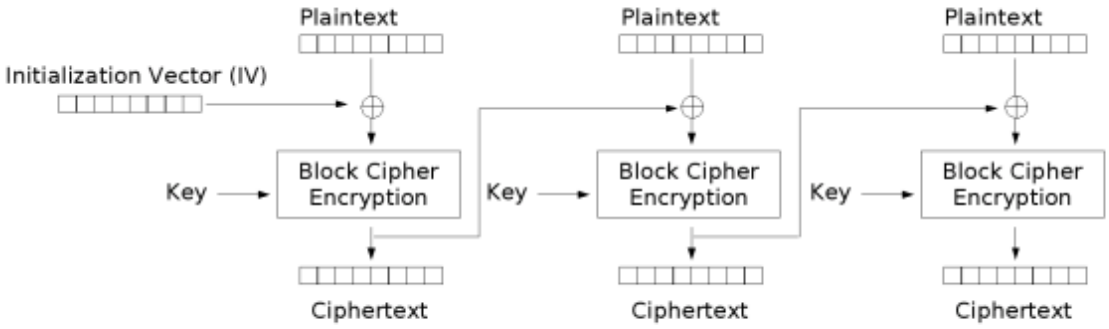
- Most algorithms operate on a fixed size of block
- For larger messages we split it up into a number of blocks and encrypt each one in serial, independently
- Hence the first block is available for transmission as soon as it is encrypted
- However this is a slight weakness, since the attacker can recognise repeated patterns and infer the relationship to the plaintext

Security

Cipher Block Chaining

- In Cipher Block Chaining each block is combined with the precedeing block.
- Note that this still means the previous block may be transmitted as soon as it is ready
- Generally *XOR* is used, if we have block N of plaintext and $\{M^{N-1}\}_K$ of cipherext, then block N is encrypted as: $\{M^N\}_K = E(K, N \oplus \{M^{N-1}\}_K)$
- Upon decryption, each block is xor'ed with the preceding block, this works since xor is its own inverse
- This is intended to prevent identical portions of the plaintext from encrypting to identical portions of ciphertext
- But there is a slight weakness at the start of each stream of of blocks
- To prevent this we insert a different piece of plaintext in front of each message, known as the *initialisation vector*.

Cipher Block Chaining



Cipher Block Chaining (CBC) mode encryption

Security

Cryptographic Algorithms

- There are many well designed cryptographic algorithms such that $E(K, M) = \{M\}_K$ such that the value of M is concealed and computing K requires a brute-force attack
- Confusion Non-destructive operations such as *xor* and circular shifting are used to combine each block of plaintext with the key
  - This confuses the relationship between M and $\{M\}_K$
  - If the blocks are larger than a few characters then this defeats attempts at cryptanalysis based on character frequencies
- Diffusion There is usually repetition and redundancy in the plaintext. Diffusion is used to dissipate regular patterns that result by transposing portions of each plaintext block.

Security

TEA — Secret Key Algorithm

- k is the key of length four (64-bit integers)
- text is originally the plaintext to be encrypted, two 64-bit integers

1. delta = 0x9e3779b9, sum = 0
2. y = text[0], z = text[1]
3. for (n= 0; n < 32; n++)
4. sum += delta
5. y += ((z << 4) + k[0]) $\oplus$ (z+sum) $\oplus$ ((z >> 5) + k[1])
6. z += ((y << 4) + k[2]) $\oplus$ (y+sum) $\oplus$ ((y >> 5) + k[3])
7. text[0] = y; text[1] = z;

Security

TEA — Tiny Encryption Algorithm

- On each of the 32 rounds the two halves of the text are repeatedly combined with shifted portions of the key and each other
- The *xor* and shifted portions of the text provide *confusion*
- Shifting and swapping of the two portions of the text provide *diffusion*
- The non-repeating constant *delta* is combined with each portion of the text on each cycle to obscure the key in case it might be revealed by a section of the text which does not vary

Security

TEA — Decryption

1. delta = 0x9e3779b9, sum = delta << 5
2. y = text[0], z = text[1]
3. for (n= 0; n < 32; n++)
4. z -= ((y << 4) + k[2]) $\oplus$ (y + sum) $\oplus$ ((y >> 5) + k[3])
5. y -= ((z << 4) + k[0]) $\oplus$ (z + sum) $\oplus$ ((z >> 5) + k[1])
6. sum -= delta;
7. text[0] = y; text[1] = z;

Security

DES

- The Data Encryption Standard
- Is mostly of historical importance now since its keys are 56-bits long
- Too short to resist brute-force attack using modern hardward
- Maps a 64-bit plaintext into a 64-bit ciphertext using a 56-bit key
- The algorithm has 16 dependent stages known as *rounds*
- Algorithm was developed in 1977 and was slow on machines of the time when written in software
- However the algorithm could be implemented in hardware and was incorporated into network interface chips

Security

DES — Cracked

- In June 1997 it was succesfully cracked in a brute-force attack
- The attack was performed as part of a competition to illustrate the need for 128-bit long keys
- About 14,000 computers took part in a distributed computation to crack the 56-bit key
- The program was aimed at cracking a known plaintext/ciphertext pair, to obtain the unknown key (and then use that to decrypt new ciphertext)
- Later, the EEF developed a machine that could successfully crack 56-bit keys in around three days

Security

Triple DES

- One solution to the weakness of 56-bit keys is to simply apply the algorithm more than once with more than one key
- $E_{3DES}(K_1, K_2, M) = E_{DES}(K1, D_{DES}(K_2, E_{DES}(K_1, M)))$
- This is equivalent to the strength of a single key with a length of around 112-bits
- But it is slow since it must be applied three times
- And DES is already considered slow by modern standards

Security

IDEA

- International Data Encryption Algorithm
- Uses 128-bit keys
- A successor to DES, its algorithm is based on the algebra of groups, and has 8 rounds of *xor*, addition modulo 2^{16} and multiplication
- Like DES uses the same function for encrytion as for decryption, which is useful if it is to be implemented in hardware.
- IDEA has been analysed extensively, and no major weaknesses have been found. It is also around three times faster than the speed of DES (and hence 9 times faster than triple-DES)

Security

AES

- US National Institute for Standards and Technology invited proposals for AES (advanced encryption standard)
- The winner, the Rijndael algorithm, was selected from 21 algorithms submitted by cryptographers in 11 countries
- The cipher has variable block and key lengths, with specifications for keys with lengths 128, 192 or 256 bits to encrypt blocks with the same lengths
- The number of rounds varies from 9 to 13
- The algorithm can be implemented efficiently on a wide range of processors and in hardware

Security

Public-key Algorithms

- There are relatively few practical public-key algorithms
- They depend on the trap-door functions of large numbers to produce keys
- The keys K_e and K_d are a pair of very large numbers
- The encryption performs an operation such as exponentiation on one of them
- Decryption is a similar function using the other key.
- If the exponention uses modulo arithmetic it can be shown that the result is the same as the original value of M, so:
- $D(K_d, E(K_e, M)) = M$
- RSA is the most widely known public-key algorithm

Security

RSA

- Rivest, Shamir and Adelman, based on the product of two very large prime numbers
- Again despite extensive attempts and investigations, no flaws have been found

Security

RSA, to find a key pair K_e, K_d

1. We need to find three numbers e, d and N, the keys will be $K_e = e, N$ and $K_d = d, N$
2. Choose two large prime number P and Q both larger than 10^{100} (a googol)
   - $N = P \times Q$
   - $Z = (P - 1) \times (Q - 1)$
3. For d choose any number that is relatively prime with Z $(gcd(d, Z) = 1)$
4. To find e, solve: $e \times d = 1 \ mod \ Z$
   - So $e \times d$ is the smallest element in the series $Z + 1, 2Z + 1, 3Z + 1, \ldots$ which is divisible by d

Security

RSA

- So the function to encrypt a single block of plaintext M is
- $E'(e, N, M) = M^e \ mod \ N$
- So the largest length of M is $log_2(N)$ bits
- And to decrypt a block of text is:
- $D'(d, N, c) = c^d \ mod \ N$
- Rivest, Shamir and Adelman proved that E' and D' are mutual inverses, so $E'(D'(x)) = D'(E'(x)) = x$ for all values of P in the range $0 \leq P \leq N$
- Note that encryption requires e and N so $K_e = e, N$
- And decryption requires d and N so $K_d = d, N$

Security

RSA — Concrete Example

1. Choose P and Q as very large prime numbers
   - $P = 5$ and $Q = 11$
2. $N = P \times Q$ and $Z = (P - 1) \times (Q - 1)$
   - $N = 55$ and $Z = 40$
3. For d choose any number that is a relative prime of Z
   - $d = 7$
4. To find e solve $e \times d = 1 \ mod \ Z$
   - $41, 81, 121, 161, \ldots$
   - $e \times 7 = 161, e = 23$
5. The numerical value of a block must be less than N, so the length of a block k must be such that $2^k < N$ here we will be forced to choose $k = 5$

RSA — Concrete Example

- So to encrypt the block M with numerical value 24 using the $K_e = 23, 55$
  1. $E'(e, N, M) = M^e \ mod \ N$
  2. $E'(e, N, M) = 24^{23} \ mod \ N$
  3. $E'(e, N, M) = 55572324035428505185378394701824 \ mod \ 55$
  4. $E'(e, N, M) = 19$
- To decrypt with $K_d = 7, 55$
- $D'(d, N, c) = c^d \ mod \ N$
- $D'(d, N, c) = 19^7 \ mod \ N$
- $D'(d, N, c) = 893871739 \ mod \ 55$
- $D'(d, N, c) = 24$

I tried first with $M = 21$ but $21^{23} \ mod \ 55 = 21$

Security

RSA — Cracking

- Given that the public key K_e contains N, to figure out e and d (and hence K_d) an attacker requires to factorise N
- In our example the prime factorisation of 55 is relatively easy to figure out $5, 11$
- The attacker would therefore know Z, they wouldn't know the choice of d but could brute-force try all possibilities
- In practice of course P and Q are chosen to be $> 10^{100}$ so $N > 10^{200}$, hence factorisation of N is extremely computationally expensive
- Factorisation of a number as large as 10^{200} would take 4 billion years using the best known algorithm on a computer that performs 1 million instructions per second.
- Intel Core i7 Extreme Edition 3960X (Hex core) = 177,730 MIPS
- $(4000000000 \times 31556900)/177730000000 = 710221$
- So 710000 years

Security

RSA Challenges

- The RSA Corporation issued a challenge to factor numbers of more than 100 decimal digits
- Numbers up to 232 decimal digits (768 binary digits) have been successfully factored
- Though there is still a 212 decimal digit (704 binary digits) number which remains unfactored
- Keys as large as 2048 bits are used in some applications
- All of this security somewhat depends upon the currently known best factoring algorithms not being improved (either because it is impossible or simply because no-one figures out how)

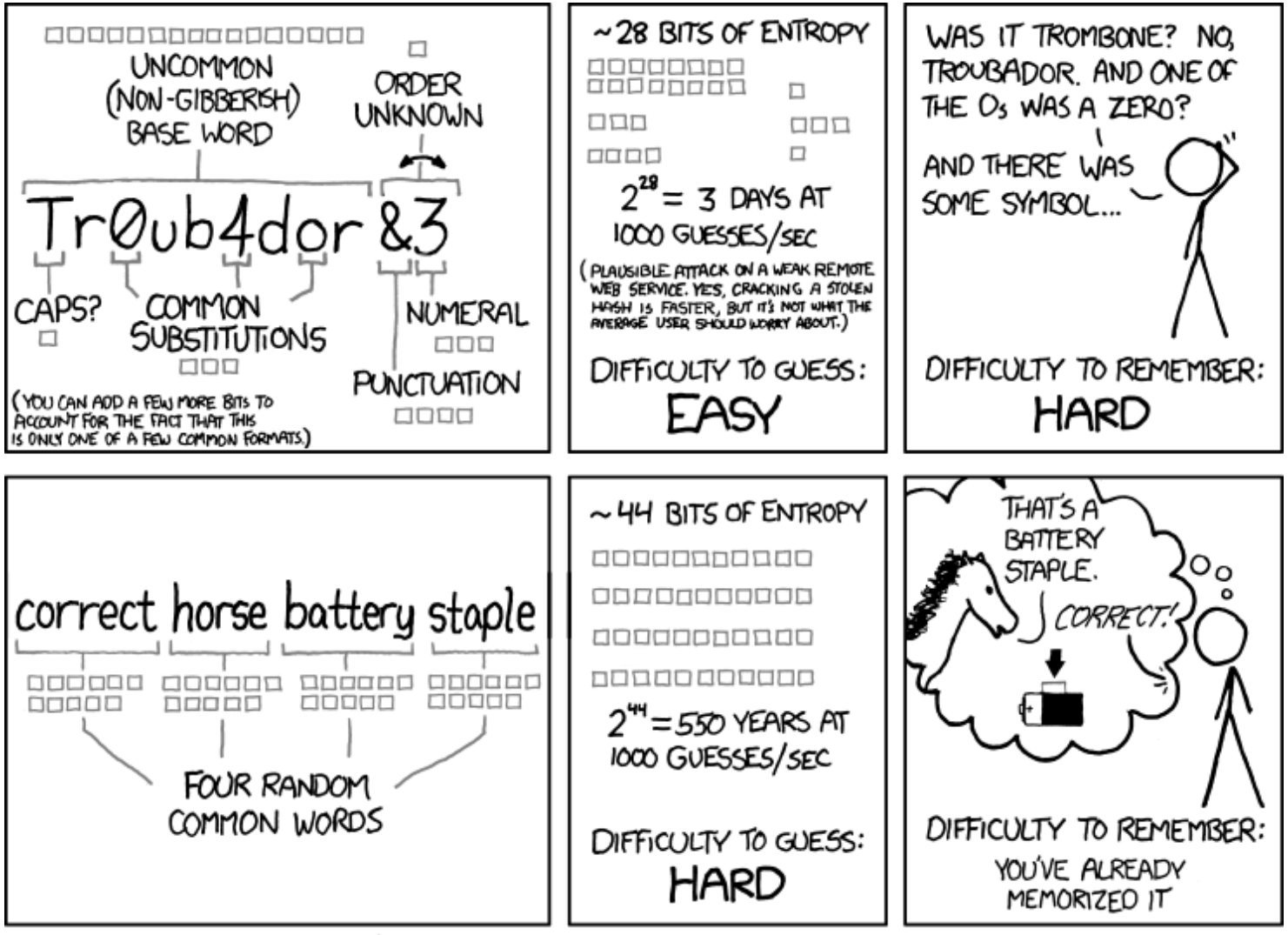
Security

Public-key algorithms

- It is worth noting a problem for all public-key cryto-algorithms
- An attacker as an unlimited supply of ciphertexts with known plaintexts
- Since the encryption is done using the public key and the attacker has access to the public key they can simply create as many plaintext/ciphertext pairs as they require
- They may even do so with any given text, for example the zero plaintext
- Additionally if the *unknown* encrypted message was really short, one could simply brute-force try all messages of the same length encrypting them to see if they match the encrypted message
  - This is obviously defeated by making sure that each message is at least as long as the key such that this form of brute-force attack is less feasible than a brute-force attack on the key

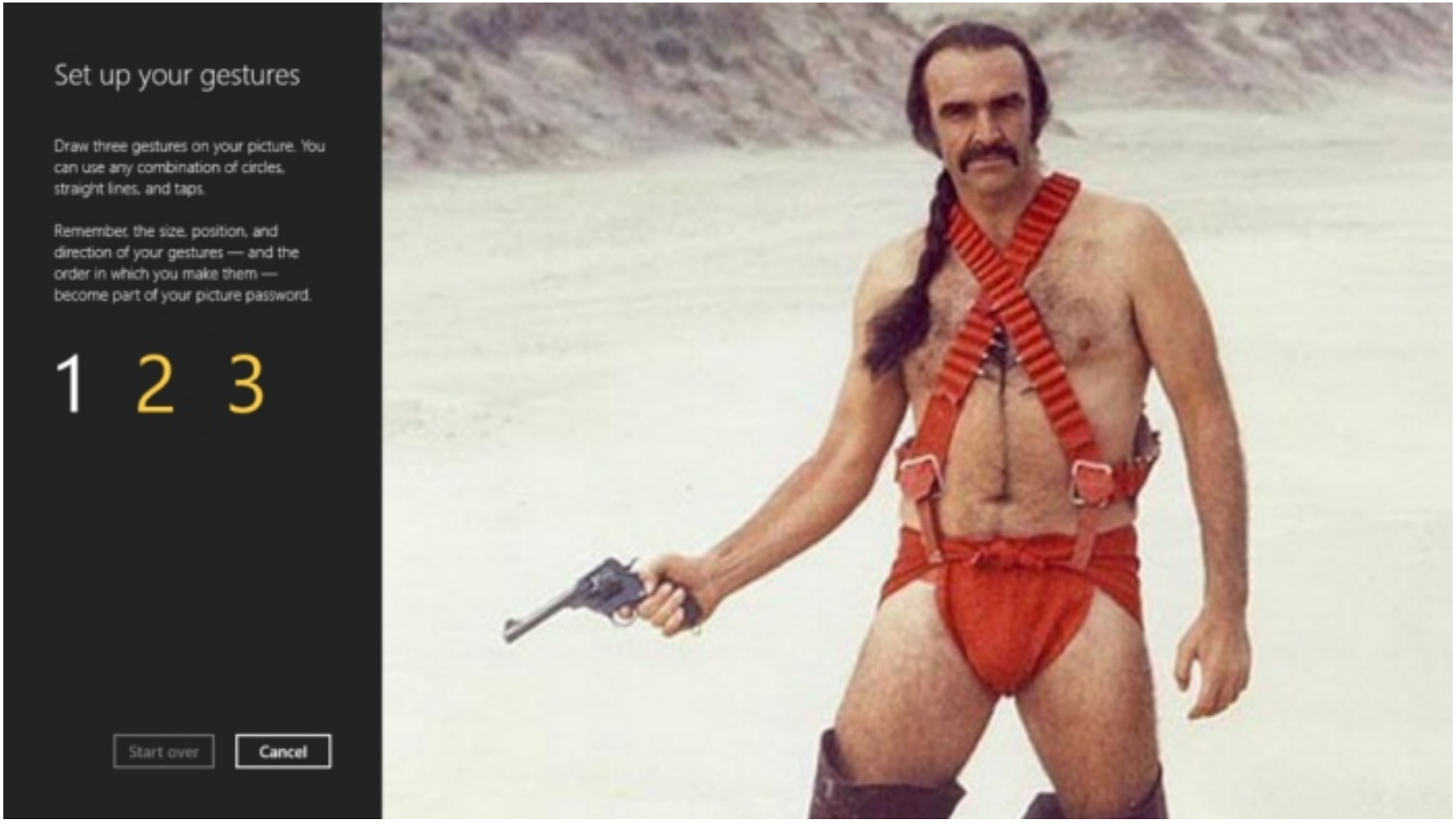
Security

XKCD



Security

Zardoz Jeff Atwood @CodingHorror recently blogged about his Surface RT authentication:



Any Questions

Any Questions?