
# Modern Cryptography

**December 2, 2005**

Matt Robshaw

France Telecom Research and Development

# Overview

- Part I: 10:00 – 11:00
  - The development of modern cryptography
- Part II: 11:30 – 12:30
  - The deployment of modern cryptography
- Part III: 14:30 – 15:30
  - Breaking cryptography and new research

# Part I: The Development of Modern Cryptology



# Overview of Part I

- Some milestones in the development of today's cryptography
- The aims of the attacker
- The goals of cryptography
- A classification of basic cryptographic primitives



# What is Cryptography?

Cryptography is about "communication in the presence of an adversary."

Rivest 1990

# What is Cryptography?

- We appeal to mathematical techniques to deal with the problems of errors
- For inadvertant corruption we use coding theory
- For malicious corruption we use cryptography



# C.E. Shannon

- The author of two classic papers
  - *A Mathematical Theory of Communication*
  - *Communication Theory of Secrecy Systems*

- Fundamental approaches to design and analysis
  - First systematic study of secrecy systems
  - Still offers guiding principles for contemporary designs

"The problem of good cipher design is essentially one of finding difficult problems."
C.E. Shannon 1946

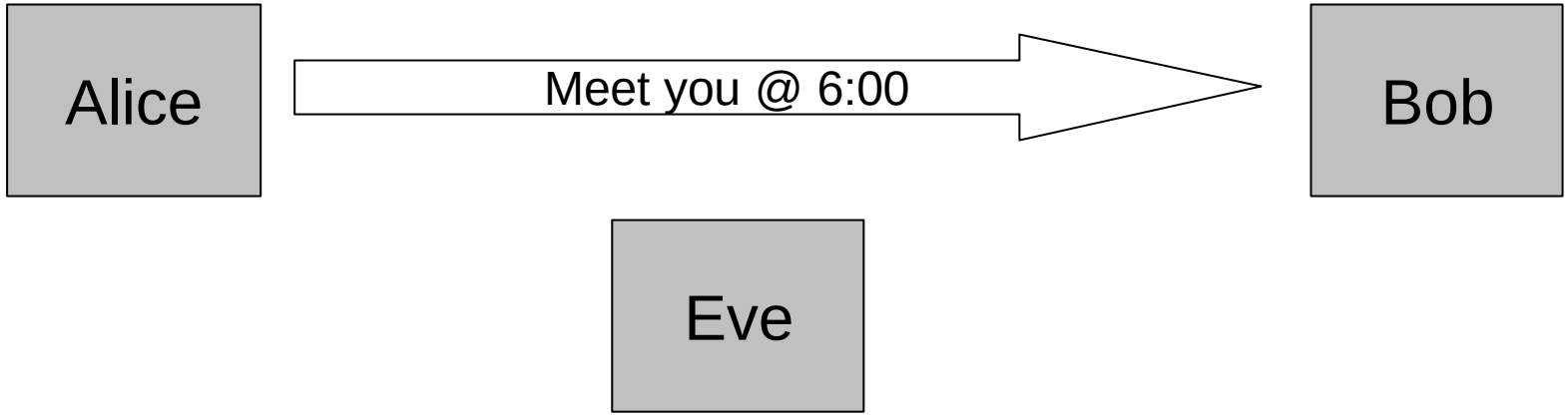
# Public Key Cryptography (PKC)

- Diffie and Hellman introduce PKC in 1976
  - Pre-'76: sender/receiver have the same key
    - symmetric cryptography
  - Post-'76: sender and receiver can have different keys
    - asymmetric cryptography
- Diffie-Hellman discussed two important concepts
  - One-way function: Easy to compute but hard to reverse
  - Trapdoor one-way function: Easy to compute but hard to reverse without some trapdoor information
- Key agreement, encryption, and digital signatures

# DES

- The Data Encryption Standard (DES) published in 1976
  - A U.S. government approved cipher
  - Designed by a non-government body (IBM)
- Promoted the widespread use of cryptography
- Provided the focus for a generation of researchers
- Together with PKC, the publication of DES turns cryptography into an open academic field

# Some Communication Threats




| Listen to message | Manipulate message | Claim a message | Spoof a message | Disavow a message |
|---|---|---|---|---|

# Some Basic Goals

- Confidentiality
  - keep content of information from unauthorized entities
- Data integrity
  - prevent unauthorized manipulation of data
- Authentication
  - **entity** authentication: establishing identity of an entity
  - **data origin** authentication: establishing the source of data
- In many applications authentication/integrity is more important than confidentiality

# The Basic Cryptographic Primitives

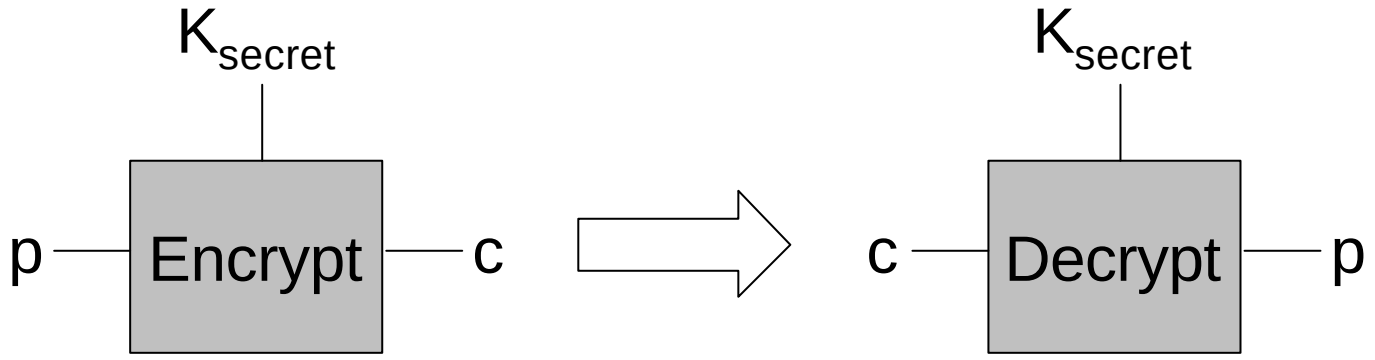
| | **Symmetric Algorithms** | **Asymmetric Algorithms** |
|---|---|---|
| **Encryption** | Block Ciphers<br>Stream Ciphers | |
| **Authentication** | | |



# Symmetric Encryption

- By some means Alice and Bob share some secret
  - This secret, or a derived value, can be used as $K_{secret}$
  - Alice and Bob can freely exchange encrypted messages
  - p is known as plaintext and c is known as ciphertext



# Block and Stream Ciphers

- Both offer symmetric encryption
  - Block ciphers operate on blocks of data
  - Stream ciphers operate on individual characters
- These ciphers have different error-propagation and synchronization properties
  - They are suitable for different environments
  - Stream ciphers can be faster than block ciphers
  - Stream ciphers can be more compact than block ciphers
  - However, block ciphers have been standardized



# The Basic Cryptographic Primitives

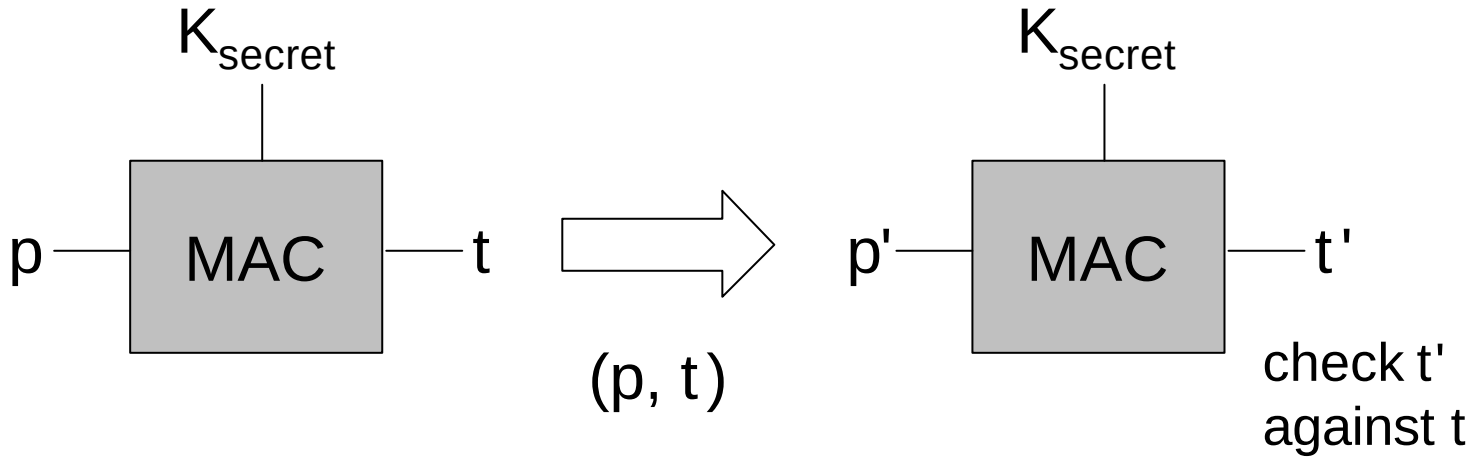
| | **Symmetric Algorithms** | **Asymmetric Algorithms** |
|---|---|---|
| **Encryption** | Block Ciphers<br>Stream Ciphers | |
| **Authentication** | Message Authentication Codes | |



# Symmetric Authentication

- By some means Alice and Bob share some secret
  - This secret, or a derived value, can be used as $K_{secret}$
  - Alice and Bob can freely authenticate messages





# The Basic Cryptographic Primitives

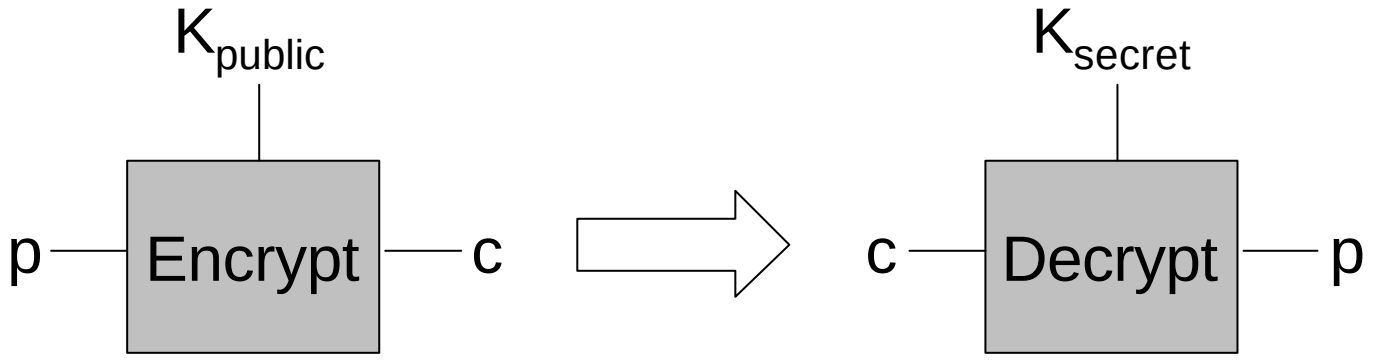
| | **Symmetric Algorithms** | **Asymmetric Algorithms** |
|---|---|---|
| **Encryption** | Block Ciphers<br>Stream Ciphers | PK Encryption |
| **Authentication** | Message Authentication Codes | |



# Asymmetric Encryption

- Alice generates a key pair of $K_{secret}$ and $K_{public}$
  - It is impossible to recover $K_{secret}$ from $K_{public}$
  - Alice publishes $K_{public}$
  - Alice can now receive messages from anyone



# The Basic Cryptographic Primitives

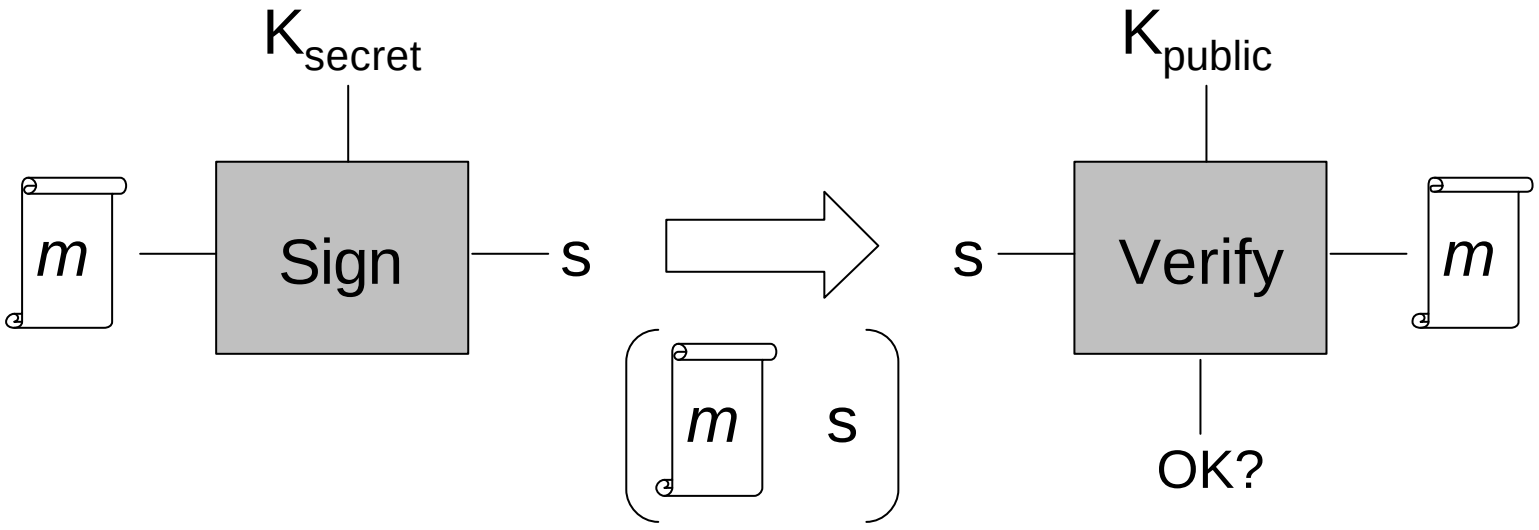
| | **Symmetric Algorithms** | **Asymmetric Algorithms** |
|---|---|---|
| **Encryption** | Block Ciphers<br>Stream Ciphers | PK Encryption |
| **Authentication** | Message Authentication Codes | Digital Signatures |



# Digital Signatures

- Alice generates a key pair of $K_{secret}$ and $K_{public}$
  - It is impossible to recover $K_{secret}$ from $K_{public}$
  - Alice publishes $K_{public}$
  - Alice can now sign a message *m*





# Algorithm Classification – By Role

**Secret Key (Symmetric)**

Block ciphers

Stream ciphers

Message authentication codes

**Public Key (Asymmetric)**

Key agreement algorithms

Public key encryption

Digital signatures

Hash function
(Message Digest Algorithm)

**Keyless**

# Choosing Cryptography

- To achieve our goals (*e.g.* encryption, authentication, *etc.*) we will need to choose a cryptographic primitive
- One issue is to decide whether we use symmetric or asymmetric cryptography
- They have different advantages and disadvantages
  - They both need a supporting infrastructure
  - The most suitable choice will be dictated by the application and environment of use



# Choosing Cryptography

- In choosing a suitable algorithm obscurity is not the same as security
- It is important that a cipher can be analysed
  - We need a good estimate of the security level
  - Simpler ciphers are almost always better
  - Complicated ciphers can rarely be trusted
- For analysis the designer requires some "structure"
  - However the cryptanalyst is also looking for "structure"
- The difficulty of design is in balancing this conflict



# Cryptographic Strength

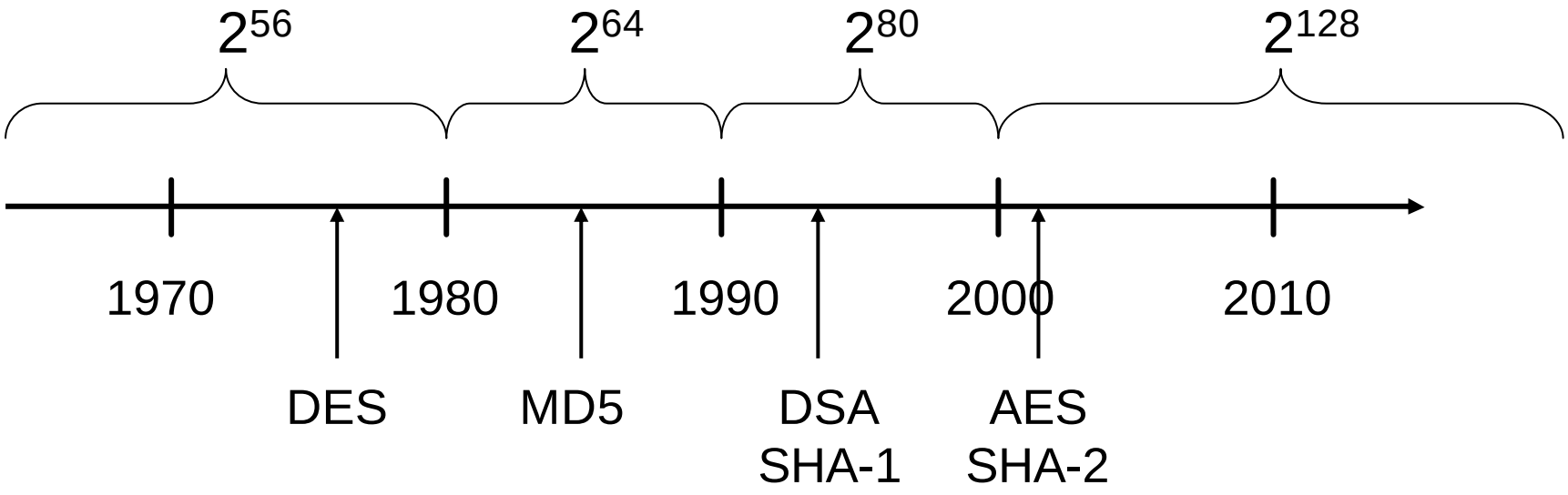
- The strength of an algorithm is measured in terms of the number of operations required to break it
  - *e.g. "To break algorithm X we need $2^{57}$ operations"*
- "Algorithm Y is intended to offer 64-bit security"
  - We expect an attack to require $2^{64}$ operations
  - This upper bound typically results from the key length
- However the real security level may be less
  - We know that an attack requires $\leq 2^{64}$ operations
  - Analysis increases our confidence that $2^{64}$ is a good estimate



# Cryptographic Strength

- An algorithm can often be dated by the intended cryptographic strength





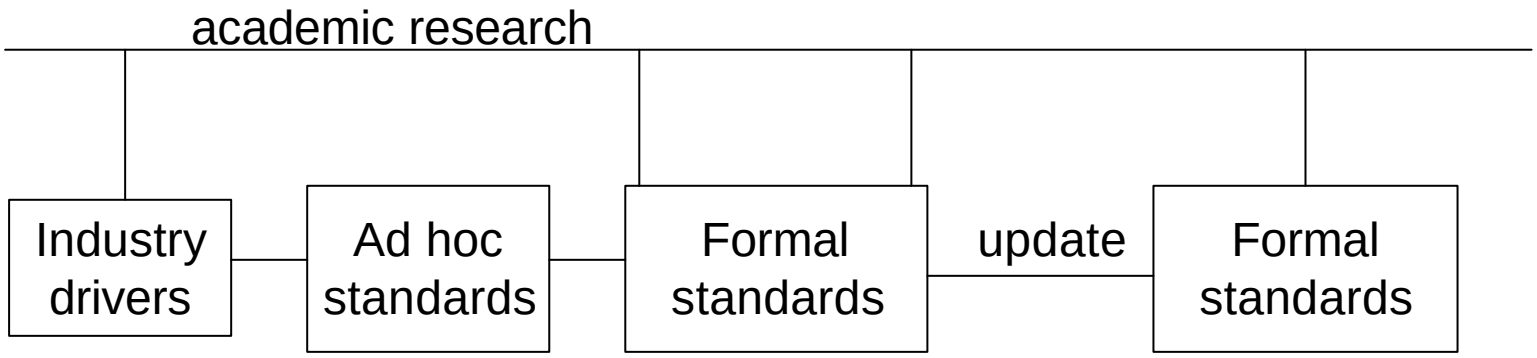
# Cryptographic Strength

- Modern cryptographic systems should provide an exceptional level of security

| | |
|---|---|
| One million | $2^{20}$ |
| Seconds in a year | $2^{25}$ |
| Global population | $2^{32}$ |
| Age of universe (years) | $2^{34}$ |
| Age of universe (μs) | $2^{80}$ |
| Protons in the universe | $2^{256}$ |

# Developing Cryptography

- Standards are vitally important
  - Unambiguous specifications of particular ciphers
  - They promote interoperability and trust
  - Standards bodies might be industry or nationally organised
  - NIST, ANSI, ISO, and IETF are among the important bodies



# Cryptographic Diversity

- When there are no clear solutions to hand everyone has an alternative
  - Hard to get consensus and interoperability
- Standards tend to produce a deployed monoculture
  - At some risk of cryptanalytic advance
- Perhaps just four algorithms are responsible for the vast majority of cryptographic use worldwide



# End of Part I

- A quick review of some important milestones in modern cryptography
- Separation of the concepts of authentication and encryption
- An introduction to the idea of symmetric and asymmetric cryptography
- In Part II, we will look at some of the technical details

# Part II: The Deployment of Modern Cryptology

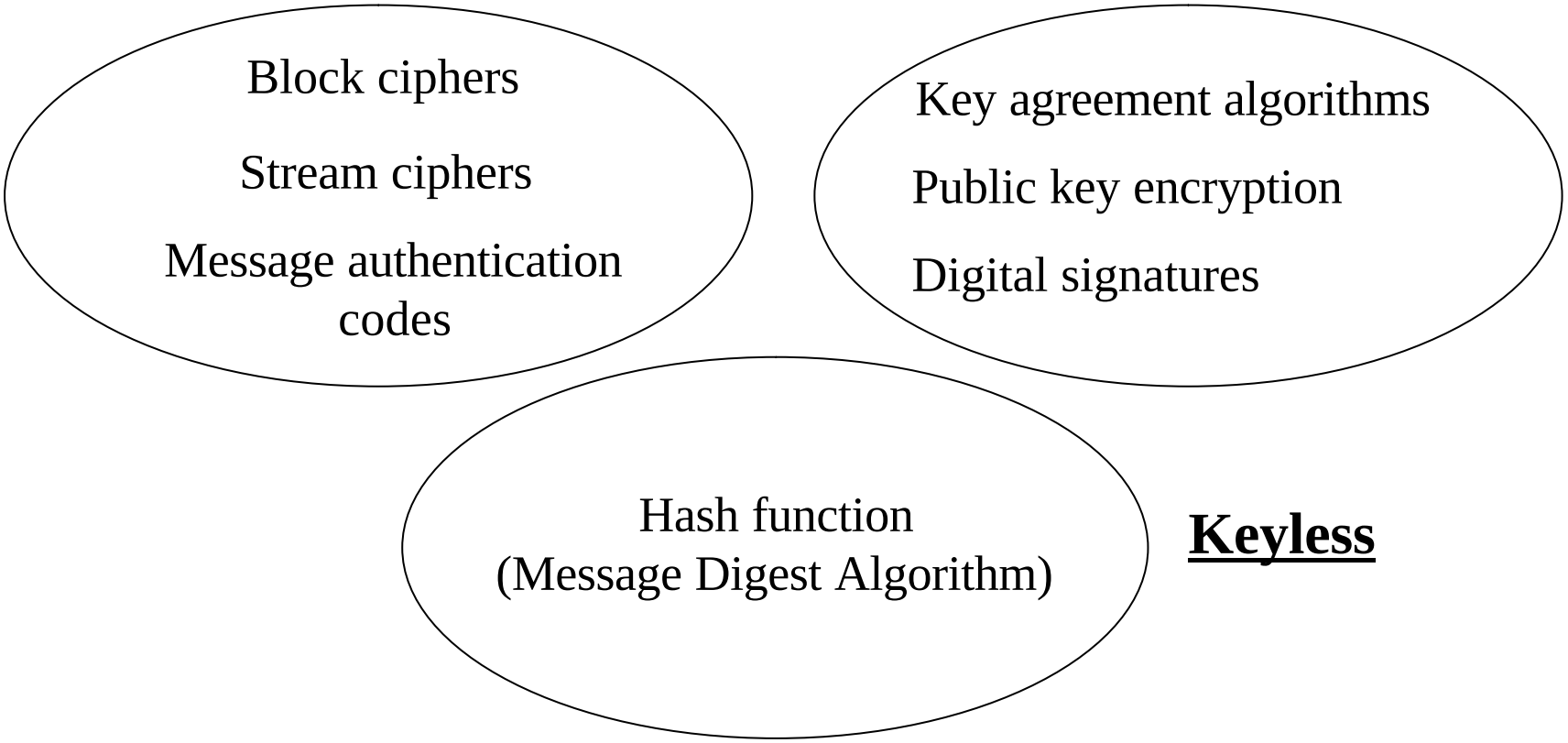
# Overview of Part II

- Symmetric cryptography
  - Block ciphers (DES, 3DES, AES)
  - Stream ciphers
  - Hash functions
- Asymmetric cryptography
  - RSA
- Cryptography in deployment



# Algorithm Classification – By Key



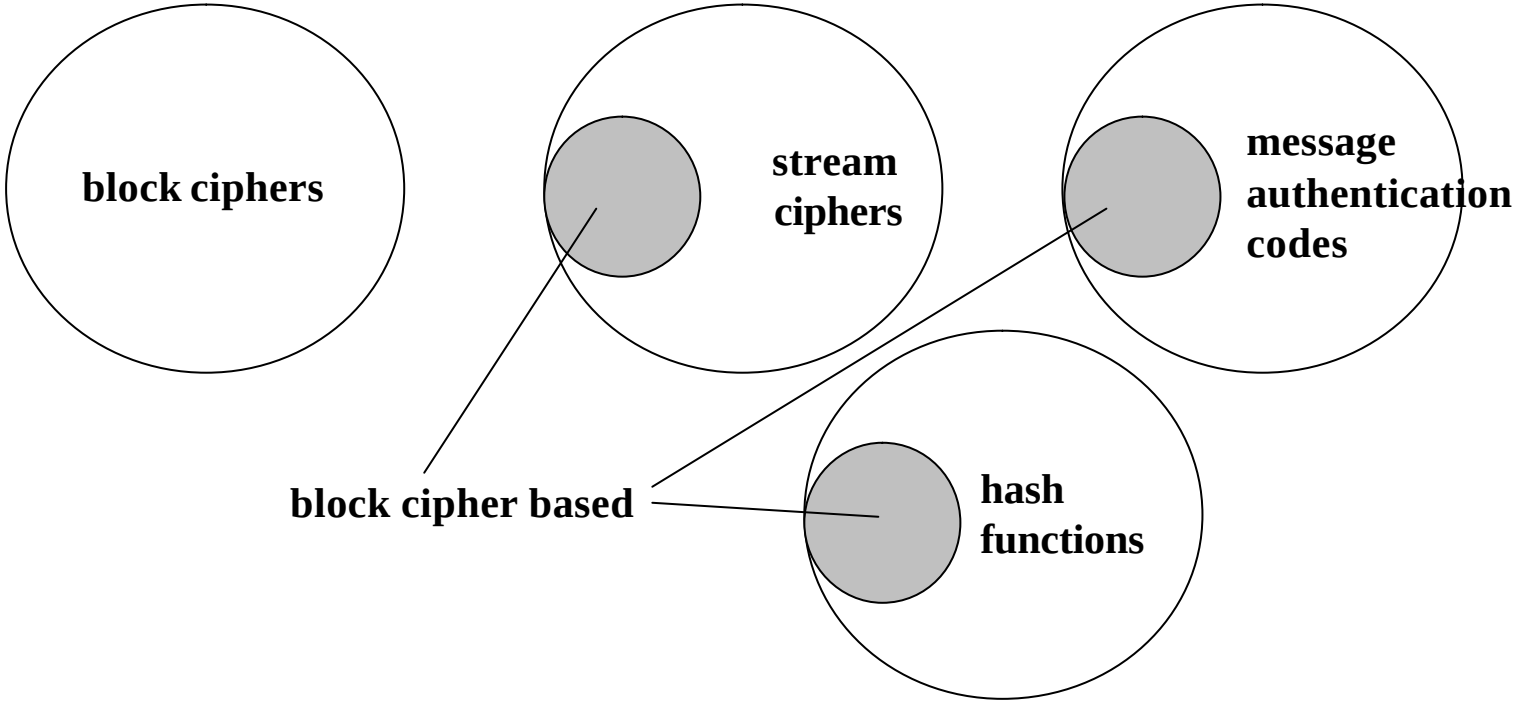


# Block Ciphers

- The most important symmetric primitive is the block cipher
- We can build the other symmetric primitives out of a block cipher
- We have a long-trusted example (DES) and trusted successors (3DES and AES)
  - Most implementations will use these standard ciphers

# Block Ciphers





# Block Ciphers

- A plaintext input *p* is a block of *b* bits
  - The block size *b* depends on the cipher (typically 64 and 128)
  - The key size s depends on the cipher (usually 128 today)
  - When using key *k* the ciphertext *c* is given by $Enc_k(p)=c$
- When a user chooses key *k*
  - The transformation of every input block is determined
  - Every input block is mapped to one (and only one) output block
  - *i.e.* we have a permutation of the $2^b$ possible input blocks
- When we change the key we change the permutation
  - In effect, a block cipher gives an indexed family of permutations

# Block Cipher Example (*b*=*3*, s =*1*)

| ***Plaintext*** | ***k = 0*** | ***k = 1*** |
|---|---|---|
| **000** | **011** | 110 |
| **001** | **101** | 010 |
| **010** | **000** | 100 |
| **011** | **010** | 001 |
| **100** | **100** | 000 |
| **101** | **001** | 111 |
| **110** | **111** | 101 |
| **111** | **110** | 011 |

# Block Cipher Example (*b*=*3*, s=*1*)

| ***Plaintext*** | ***k = 0*** | ***k = 1*** |
|---|---|---|
| **000** | **011** | 110 |
| **001** | **101** | 010 |
| **010** | **000** | 100 |
| **011** | **010** | 001 |
| **100** | **100** | 000 |
| **101** | **001** | 111 |
| **110** | **111** | 101 |
| **111** | **110** | 011 |

# DES (Data Encryption Standard)

- Published in 1976 by NBS later renamed NIST
  - National Institute of Standards and Technology
  - Sets Federal Information Processing Standards (FIPS)
- DES was regularly re-affirmed until 1998
  - No longer recommended
- The best practical attack is brute-force key search
  - This requires $2^{56}$ operations (which is feasible)
- There are mathematical attacks
  - But these require around $2^{43}$ plaintexts

# DES Origins

- May 1973
  - Call for proposals with disappointing results
- August 1974
  - Second call for proposals
- IBM were successful with a variant of *Lucifer*
- August 1976
  - Draft standard published

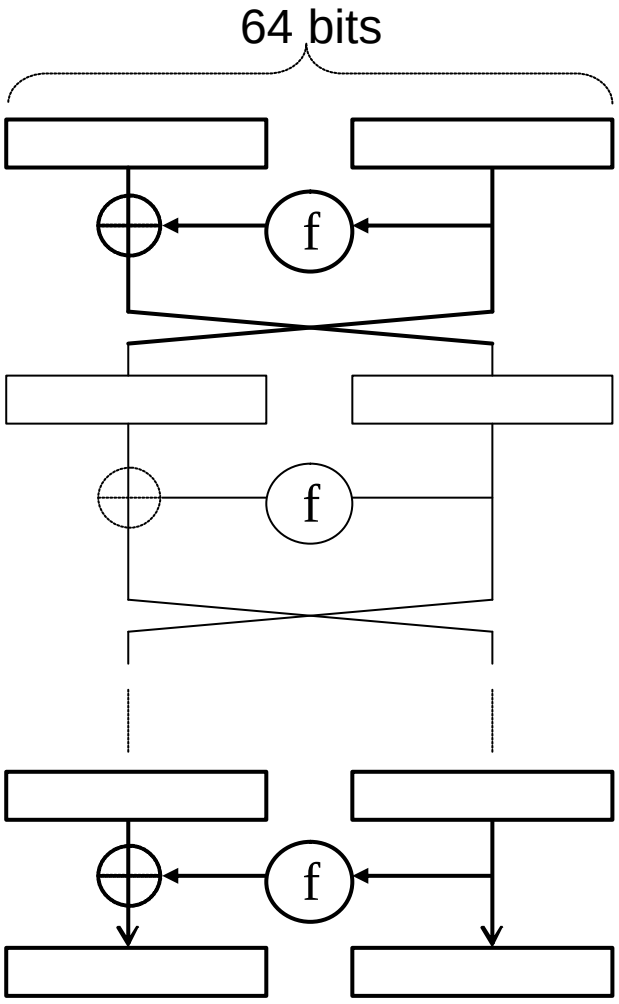
# Features of DES

- Built very closely on design principles of Shannon
  - Other design decisions were kept secret until 1994
- Designed around the manipulation of bits
  - Very good performance in hardware
  - Not so good in software
- DES operates on 64-bit blocks with a 56-bit key
- DES was incorporated into many other standards
  - Widely-used, particularly in the financial industries

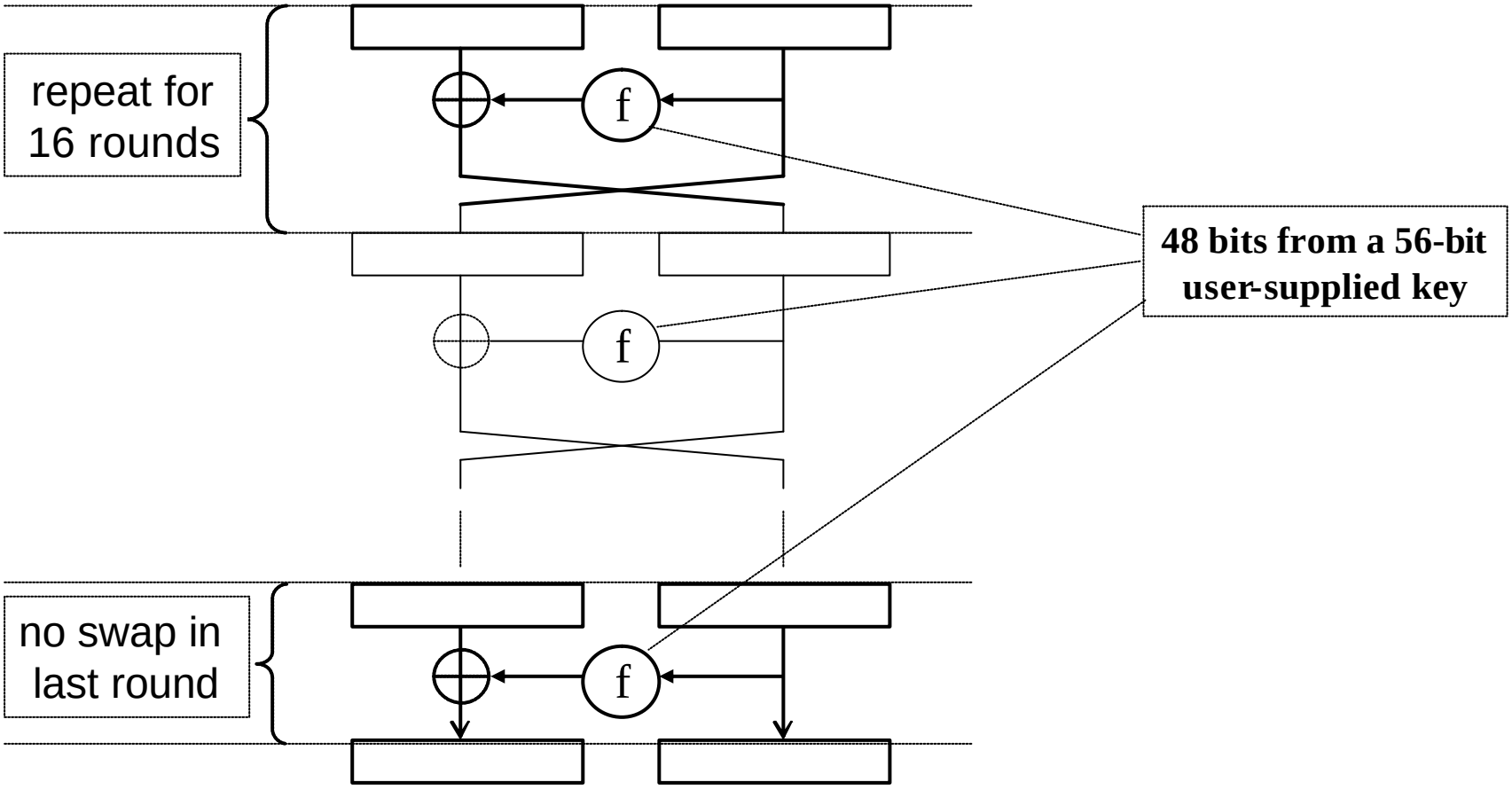


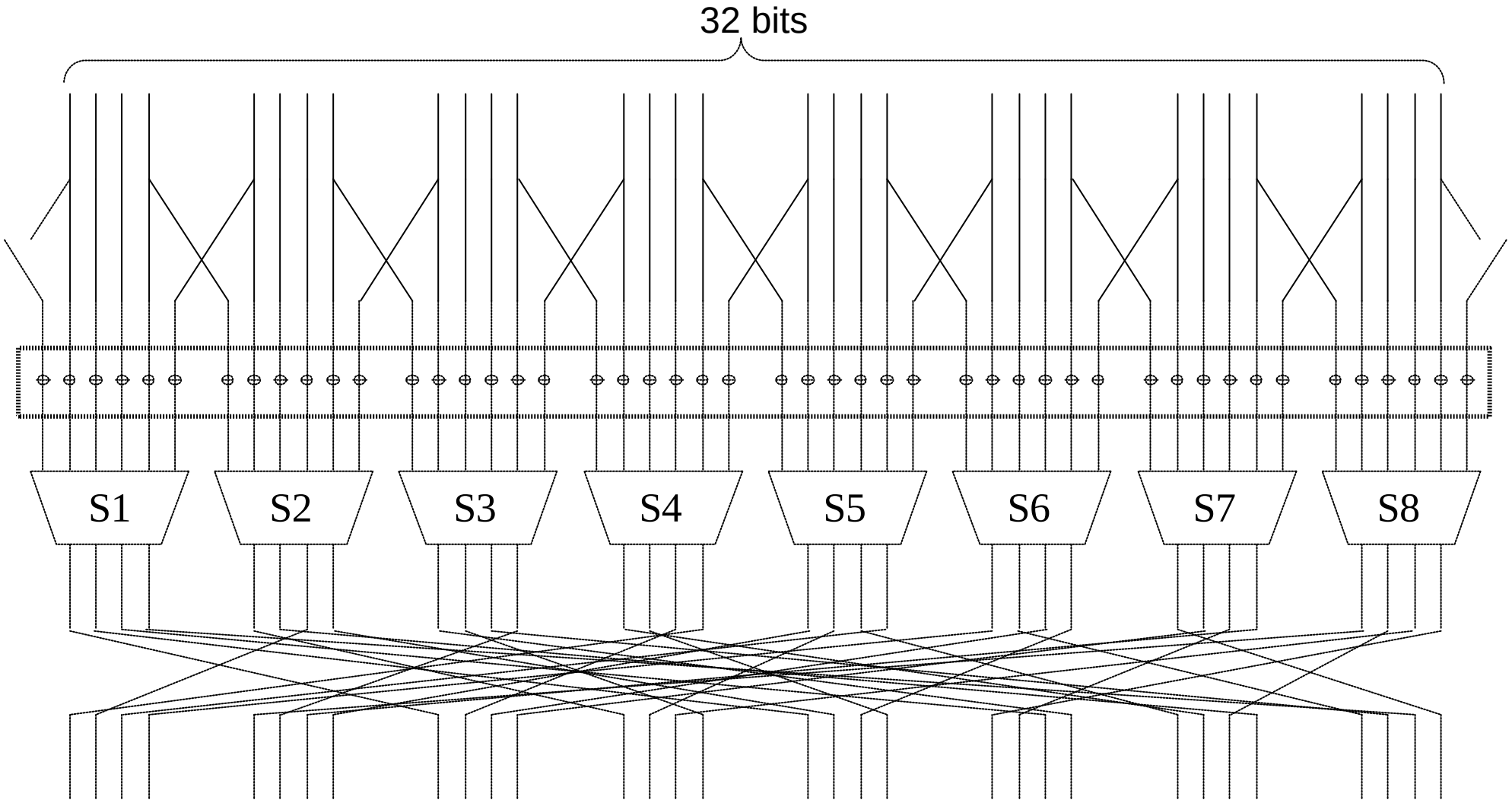
# Basic Structure of DES



# Basic Structure of DES



# DES Round Function



# DES Reception

- The initial reaction was one of suspicion
  - Why is the key only 56 bits long?
  - The S-boxes are vital for the security of DES
    - What was the role of the NSA in the choice of S-boxes?

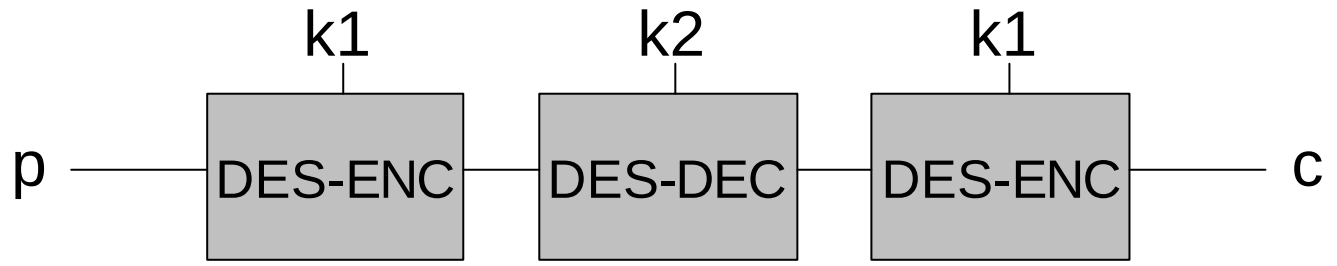
"Structures have been found in DES that were undoubtedly inserted to strengthen the system against certain types of attacks."

Lexar Report 1976

- DES design principles were released in 1994
  - The designers used differential cryptanalysis prior to 1976
  - The attack was discovered in the open community in 1989

# 3DES (Triple-DES)

- DES is no longer recommended
- However a cipher built on DES is used widely
  - It consists of three iterations of DES, *e.g.*




  - This variant is referred to as 2-key EDE Triple-DES (3DES)
  - 3DES is widely used in banking and financial industries

# Advanced Encryption Standard

- Finalised in October 2000 as a replacement to DES
  - The AES will also replace 3DES (eventually)
- An open competition was used to find the AES
  - Required four years, 15 entries from around the world
  - The candidate Rijndael was chosen as the final AES
- Larger parameters than DES
  - 128-bit blocks and key sizes of 128, 192, and 256 bits
  - More flexible performance profile than DES and 3DES

**csrc.nist.gov/CryptoToolkit/tkencryption.html**

# Block Ciphers in Practice

- The trusted alternatives are 3DES and AES
  - It is easier to upgrade DES-infrastructures to 3DES
  - New deployments will probably use AES
- Some block ciphers that you might see include
  - Blowfish in assorted internet applications
  - IDEA in PGP
  - KASUMI in 3rd generation mobile systems
  - RC5 in some mobile phone applications
  - SAFER in Bluetooth

# The Vernam Cipher (1917)

- Gilbert Vernam developed an unbreakable cipher
- Encode a message as a series of 0's and 1's
- Flip each bit independently with probability ½

| | |
|---|---|
| Message | 1 0 0 0 1 1 0 0 0 1 1 |
| Flipping sequence | 1 1 0 1 0 1 0 1 0 1 0 |
| Encrypted Message | 0 1 0 1 0 0 0 1 0 0 1 |



# The Vernam Cipher (1917)

- An eavesdropper might intercept 01010001001
  - Without the "flipping sequence" it is impossible to decrypt
- But there are some problems(!)
  - How do we get the "flipping sequence" from the sender to the intended receiver?
  - A "flipping sequence" cannot be compressed and so it must be as long as the message
  - A "flipping sequence" can only be used once
- Despite these problems, the cipher has been used and is known as the one-time pad

# Stream Ciphers

- The random flipping sequence is not practical
  - Replace it with one that looks random – a keystream
- The keystream is generated from the secret key
  - Deciding on what makes a good keystream is not easy
  - Analysis of the keystream should not leak key information
  - Analysis should not allow prediction of the next bit
- During the 1960's a very elegant mathematical framework for design and analysis was established
  - Built around linear feedback shift registers (LFSRs)

# Stream Ciphers

- Unfortunately the same rich design structure aids the cryptanalyst
  - It is very difficult to build a secure stream cipher
- Nevertheless, stream ciphers are widely deployed
  - It is easy to change a basic configuration to make unique ciphers for particular applications
- One that is used very often is RC4 (*e.g.* in SSL)
  - A very simple cipher but not based on LFSRs

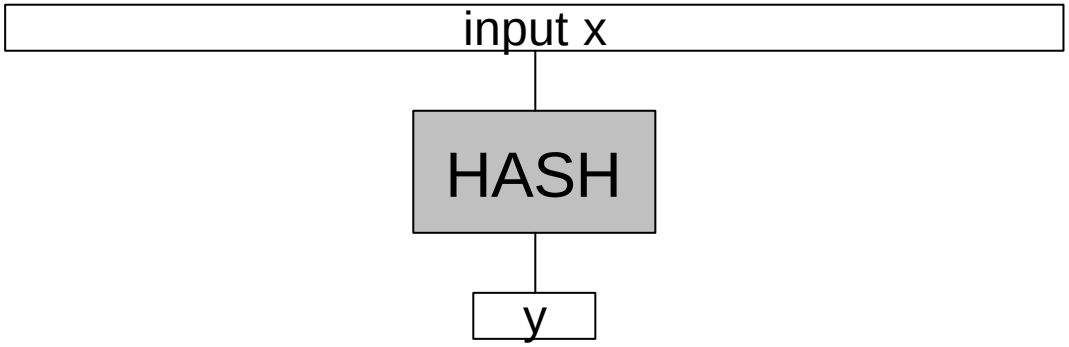
# Stream Ciphers in Practice

- There is no DES-equivalent for stream ciphers
  - The field of stream ciphers is very fragmented
- eSTREAM is an EU-supported research effort
  - New stream ciphers have been invited for analysis
  - Intended to be fast in software or compact in hardware
  - 34 new proposals were submitted in April 2005
  - Currently around ½ have security concerns

    **www.ecrypt.eu.org/estream**
- Today the benchmark stream cipher would perhaps be one based on the AES

# Hash Functions

- A keyless, but very useful, primitive
  - A hash function takes an input of arbitrary length
  - Produces a random-looking output of a fixed length
  - Typical output lengths are 128, 160, or 256 bits



# Hash Function Properties

- Given output *y* it is hard to find an input *x* with *h(x)=y* (finding a pre-image**)**
- Given an output *y* and an input *x* with *h(x)=y*, it is hard to find a second input *x′ ? x* with *h(x′)=y* (finding a 2nd pre-image**)**
- It is hard to find any inputs *x* and *x′* with *h(x)= h(x′)* (finding a collision**)**

# Hash Functions In Practice

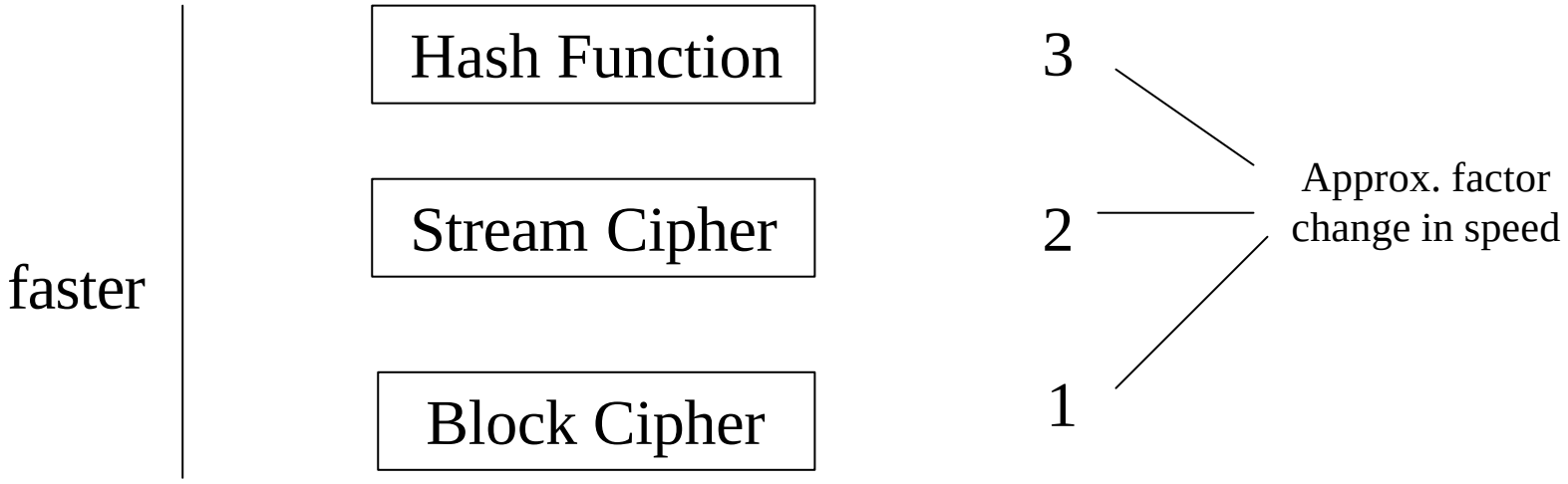
| Ideally … | **Pre-image** | **2nd Pre-image** | **Collision** |
|---|---|---|---|
| Operations to break an n-bit hash function | $2^n$ | $2^n$ | $2^{n/2}$ |

- Essentially two hash functions are in common use
  - MD5 and SHA-1
  - The collision-resistance of both has been compromised
- New hash functions are being developed
  - But a new standard will take between 5 and 10 years

**csrc.nist.gov/CryptoToolkit/tkhash.html**

# A Performance Guide

- A reasonable(?) rule of thumb



# Symmetric Cryptography

- Block ciphers are currently in good shape
  - The state of stream ciphers and hash functions is questionable
- The cryptography currently deployed remains sound
  - Attacks on some deployed algorithms don't yet seem to threaten the implementations
- But some new stream ciphers and hash functions are desperately needed
  - These will be one of the most active areas of research in the coming years

# Asymmetric Cryptography

- Just as for symmetric cryptography there are algorithms for confidentiality and authentication
  - Confidentiality is given by an encryption algorithm
  - In the public-key setting, authentication is provided by a digital signature
- The classic DH-way to make a digital signature scheme is to start from an encryption algorithm
  - Such signature schemes are called reversible
- But not all signature schemes are derived like this

# Asymmetric Cryptography

- There is a third class of asymmetric algorithm
- So-called key agreement algorithms
  - These allow two parties to agree on a common secret key
  - An example was provided in the "New Directions" paper
- But key agreement can also be achieved with an encryption algorithm
  - Key agreement algorithms are used less often

# Asymmetric Cryptography

- With symmetric cryptography we had a significant problem
  - "By some means Alice and Bob agree on a shared secret"
- However with a public encryption key, anyone can send a message to Alice without having met
- Thus asymmetric cryptography solves one problem
  - However, it raises others and establishing what is termed a public key infrastructure (PKI) is hard

# Asymmetric Cryptography

- Setting up a secret and public key requires special building blocks
- These are constructed from hard mathematical problems
- Three hard problems are commonly used
  - Factoring
  - Discrete logarithms
  - Elliptic curve discrete logarithms

# Hard Problems

- All three problems have their proponents!
  - For integer factorisation (IF) and discrete logarithms (DL) we can use very similar algorithms
  - Elliptic curve discrete logarithms (ECDL) are much harder
  - Thus, for similar security, ECDL can use smaller parameters
- Factoring a 1024-bit RSA number is roughly as difficult as solving a 160-bit ECDL problem
  - Can lead to smaller keys and reduced storage/bandwidth
- Currently most deployments are IF based

# A Basic Classification

| | Factorization | Discrete Logarithm | EC Discrete Logarithm |
|---|---|---|---|
| Key Exchange | | Diffie-Hellman | ECDH |
| Encryption | RSA | El-Gamal | ECIES |
| Signature | RSA | DSA | ECDSA |

roughly same size keys

smaller keys

# Factoring as a Hard Problem

"There are many cases in which we can easily and infallibly do a certain thing but may have much trouble undoing it … Given any two numbers, we may by a simple and infallible process obtain their product, but when a large number is given it is quite another matter to determine its factors. Can the reader say what two numbers multiplied together will produce the number 8,616,460,799? I think it is unlikely that anyone but myself will ever know; for they are two large prime numbers."

W.S. Jevons 1873

# Factoring as a Hard Problem

- What are the factors of this 309-digit number?

  135066410865995223349603216278805969938881475605667027524485143851526510604859533833940287150571909441798207282164471551373680419703964191743046496589274256239341020864383202110372958725762358509643110564073501508187510676594629205563685529475213500852879416377328533906109750544334999811150056977236890927563

- There is a reward of $100,000

# RSA (1979)

- Named after Rivest, Shamir, and Adleman
- The most widely-used asymmetric algorithm
- If you can efficiently factor you can break RSA
  - However it is unknown whether it is possible to break RSA without factoring
- An exceptionally simple and elegant algorithm

# RSA Set-Up

- Alice generates primes $p$, $q$ and computes $N = pq$
- Alice chooses a public exponent $e$
  - $e$ must be co-prime to $(p-1)(q-1)$
- Alice generates secret exponent $d$ so that $ed = 1$ modulo $(p-1)(q-1)$
  - *i.e.* so that $(p-1)(q-1)$ divides $ed-1$
- The public key for Alice is the pair $(N, e)$
  - The secret key is $d$ (also primes $p$ and $q$)

# RSA Encryption

- Alice publishes the public key (*N*, *e*)
- Bob wishes to send message *m* to Alice
- Bob computes $c = m^e$ modulo *N*
  - i.e. computes the remainder when $m^e$ is divided by *N*
- Bob sends *c* to Alice

# RSA Decryption

- Alice receives $c$
- Alice computes $m' = c^d$ modulo $N$
  - i.e. computes the remainder when $c^d$ is divided by $N$
- The mathematics of RSA guarantees that $m'=m$

# RSA Signatures

- Alice publishes the public key (*N*, *e*)
- Alice wishes to sign message *m*
  - She computes *h=Hash(m)* to act as a representative for *m*
- Alice computes $s = h^d$ modulo *N*
  - i.e. computes the remainder when $h^d$ is divided by *N*
  - Note that *d* is the secret exponent
- *s* is the signature of *m* and can be sent as (*m*, *s* )

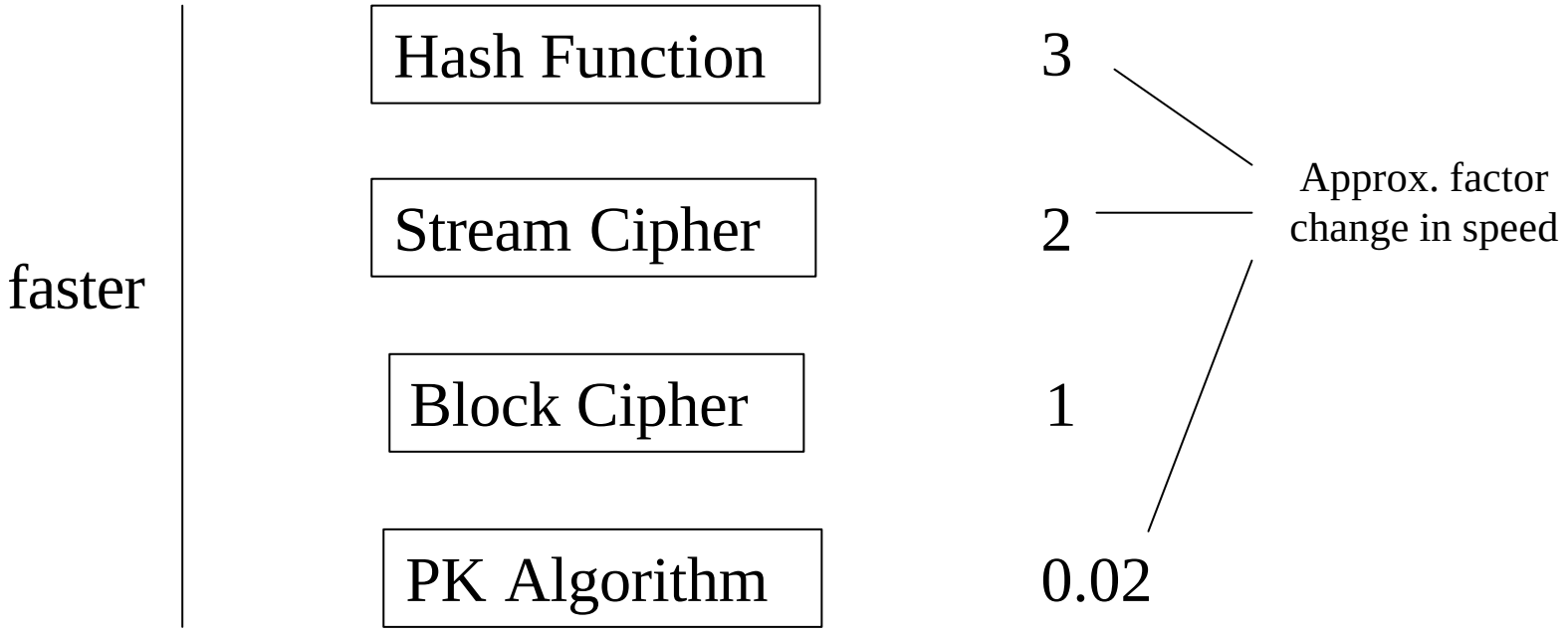
(NB: RSA offers message recovery and *s* can contain short messages *m*)

# RSA Signature Verification

- Bob receives the signed message (*m, s* )
- Bob recovers the public key (*N, e* )
- Bob computes $h' = s^e$ modulo *N*
  - i.e. computes the remainder when $s^e$ is divided by *N*
- If *h' = Hash(m)* the mathematics of RSA assures Bob that *s* was generated using *d*
  - i.e. the signature could only have been created by someone possessing *d* (namely Alice)

# Why not use PK everywhere?

- A reasonable(?) rule of thumb



# Hybrid Solutions

- RSA is widely used
  - But compared to symmetric cryptography RSA is slow
- For encryption use a block or stream cipher
  - Use RSA encryption to share the symmetric key *k*
    - Take Alice's public (*N,e*) and compute $c = k^e$ modulo *N*
    - Send *c* to Alice
- For signing we sign the hash of a message
  - Use the hash function to compress the message *m*
    - Use RSA to sign *Hash(m)* instead of *m*

# Asymmetric Cryptography

- Currently RSA provides the majority of deployed asymmetric cryptography
- For digital signatures an important alternative is DSA
  - NIST Digital Signature Algorithm
  - Based on the discrete logarithm problem
- We may begin to see more ECDL deployments, but at the moment they are minimal

# Applications are Rarely Simple

- Many modern applications use a complex mix of cryptographic algorithms
  - *e.g.* the SSL-handshake uses many algorithms
    - Hash function during random number generation
    - Hash function for key derivation
    - Digital signatures for public key certificate
    - Public key encryption
    - Stream cipher for data encryption
    - Message authentication code for data authentication
- Need to be sure that cryptography doesn't get in the way

# End of Part II

- We have looked at different cryptographic primitives in some detail
- In Part III we will consider how to break cryptographic algorithms
- We will also look at some new research directions

# Part III: Breaking Cryptography and Future Directions

# Overview of Part III

- Breaking cryptography
  - Exhaustive search
  - Factoring
  - Attacks on an implementation
- Security and cryptography in perspective
- Three current research directions
- Conclusions

# Breaking Cryptography

- Essentially there are three approaches to breaking an algorithm
- Brute force attacks
  - Exhaustive key search or solving the hard problem
  - These attacks can always be mounted
- Mathematical analysis
  - For a good algorithm mathematical analysis should be infeasible
- Side-channel cryptanalysis of the implementation

# Exhaustive Key Search

- This is the most basic attack
  - It can always be attempted
  - If the best way to attack a cipher is exhaustive search then the cipher should arguable be considered a "good" cipher
  - The practical security depends on the number of keys

| | |
|---|---|
| $2^{40}$ operations | Easy |
| $2^{64}$ operations | Practical |
| $2^{80}$ operations | Not yet feasible |
| $2^{128}$ operations | Very strong |

# DES Exhaustive Key Proposals

- Various search efforts have been proposed
  - The time given is to exhaust the DES key space ($2^{56}$)

| | Year | Cost | Time |
|---|---|---|---|
| Diffie+Hellman | 1977 | $20,000,000 | 20 hours |
| Wiener | 1993 | $1,000,000 | 7 hours |
| BSA | 1995 | $300,000 | 6 hours |
| Wiener | 1998 | $1,000,000 | 70 min. |
| Distributed.net | 1997 | ≈ $0 | 140 days |
| EFF | 1998 | $250,000 | 9 days |
| Distributed.net+EFF | 1999 | $250,000 | 3 days |

- The last three proposals were implemented

# DES Cracker

- EFF built a DES cracker
  - Consisted of 37,000 specially designed search units
  - Required around $250,000 in 1998
  - DES cracker tested around $92 \times 10^9$ keys/sec

# Types of Exhaustive Key Search

- Distributed search in software
  - The start-up cost of the Internet-based search is low
  - The power of the Internet-based search is initially low, but grows massively with the number of participants
  - Distributed search over the Internet likely to be open
- Dedicated hardware design (*e.g.* DES cracker)
  - Start-up costs are high
  - The hardware-based search runs at a constant rate
  - A hardware-based search can be hidden
- Both tackle the search problem in parallel

# Distributed.Net

- RSA Data Security launched challenges in 1997
  - Challenges to recover 40- to 128-bit keys
- Distributed.net attempts to harness massive amounts of computing power
  - 64-bit key found in July 2002 after testing 15,769,938,165,961,326,592 keys using 58,747,597,657 computing units

| | Time (days) | % of Keyspace | Complete Search |
|---|---|---|---|
| 64-bit | 1,757 | 85 | 5.6 years |

  - The 72-bit challenge is ongoing

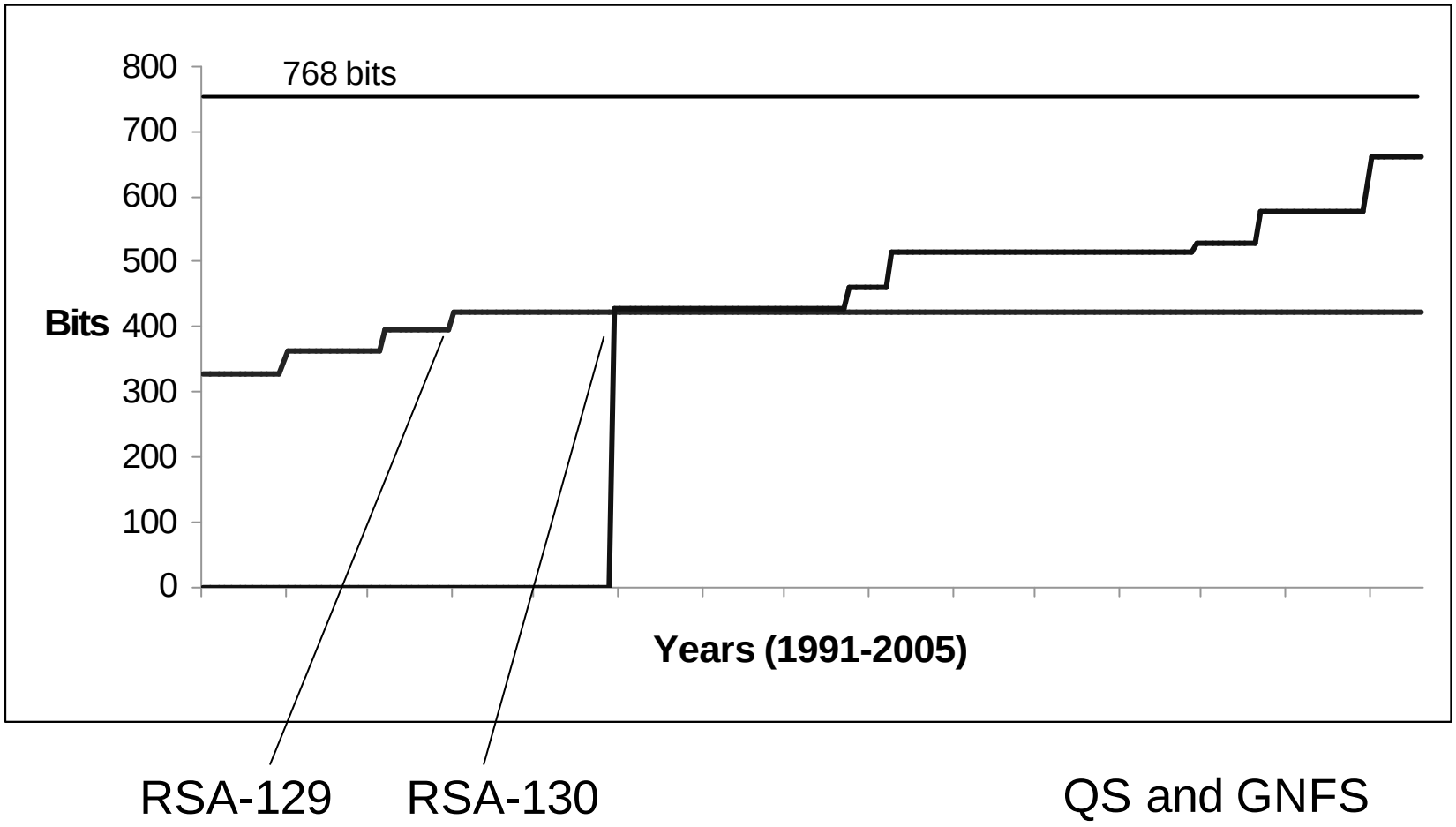
# RSA Factoring Challenge

- RSA Security tracks the state of factoring ability by means of a challenge
- Researchers are encouraged to factor increasing large RSA-like numbers
  - Factoring is far more sophisticated than exhaustive search but we might still view it as a brute-force attack
- These factorisations provide important data points
  - Allow industry to set and update security recommendations
  - Current recommendations are for 768-, 1024-, and 2048-bit *N*

# Recent Factoring Achievements

| Number | Digits | Bits | Date |
|---|---|---|---|
| **RSA-129** | **129** | **426** | **April 1994** |
| RSA-130 | 130 | 430 | April 1996 |
| RSA-140 | 140 | 463 | February 1999 |
| RSA-150 | 150 | 496 | April 2004 |
| **RSA-155** | **155** | **512** | **August 1999** |
| RSA-160 | 160 | 530 | March 2003 |
| RSA-576 | 174 | 576 | December 2003 |
| RSA-640 | 193 | 640 | November 2005 |
| RSA-200 | 200 | 663 | May 2005 |

# Factoring Progression



# Factoring Algorithms

- Current best-factoring techniques for RSA moduli involve two phases
  - A relationship gathering phase
  - A matrix reduction phase
- The first phase can be distributed across the Internet
  - The second cannot be distributed and is a bottle-neck
- The General Number Field Sieve is used today
  - The Special Number Field Sieve cannot currently be applied to RSA moduli but has a better running time

# Unclaimed Factoring Prizes

| Number | Digits | Bits | Prize |
|---|---|---|---|
| RSA-704 | 212 | 704 | $30,000 |
| RSA-768 | 232 | 768 | $50,000 |
| RSA-896 | 270 | 896 | $75,000 |
| RSA-1024 | 309 | 1024 | $100,000 |
| RSA-1536 | 463 | 1536 | $150,000 |
| RSA-2048 | 617 | 2048 | $200,000 |

# Mathematical Analysis

- Cryptanalysts will spend an inordinate amount of time trying to beat brute-force attacks
  - An academic "break" can be as slight as $2^{126}$ operations for an algorithm that is claimed to have 128-bit security
  - Such results may have little or no practical impact
  - But they might still be viewed as a "certificational" weakness
- We trust ciphers more when we believe that they have resisted the attentions of many cryptanalysts
  - This is not always easy to gauge
  - Depends on the prominence and simplicity of the algorithm

# Some Basic Assumptions

- Kerckhoffs' Assumption (1883)

  The security of the cryptographic systems resides solely in the secrecy of the key

- For analysis we give the adversary every advantage

# Types of Attacks on Encryption

- Ciphertext only
  - the attacker gets the ciphertext and has knowledge of the source statistics (e.g. ASCII encoded)
- Known plaintext
  - the attacker knows the values of the plaintext being encrypted (e.g. header information)
- Chosen plaintext
  - the attacker chooses the plaintext being encrypted
- For analysis we give the adversary every advantage

# Side-Channel Attacks

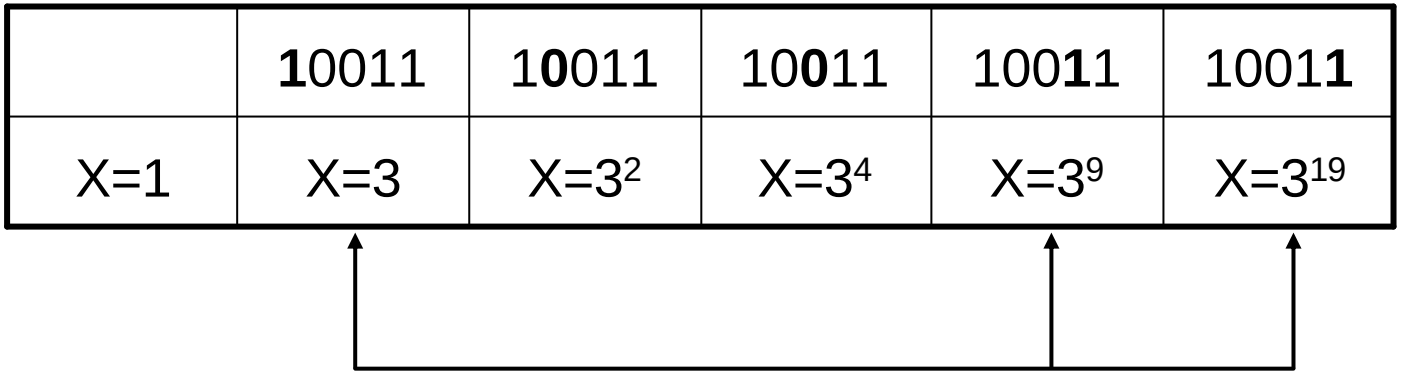
- Secret information can be leaked by physical means
  - The most well-known are variations in timing or power consumption during a cryptographic computation
- Might be applied to deployments such as smart cards
  - Potentially these might contain vulnerable implementations
  - The attacker can mount the attack in a closely controlled environment
- There are sophisticated variants with wider applicability

# Attacking RSA Implementations

- Consider the RSA signing operation:
  - Alice computes $s = h^d$ modulo $N$
- One way to exponentiate is via square-and-multiply
  - *e.g.* to compute $x = 3^{19}$
  - Write 19 in binary as 10011
  - Set x = 1 and scan the exponent from left to right
    - always square the current value of x at each step
    - multiply by 3 if the exponent bit is 1

| | **1**0011 | 1**0**011 | 10**0**11 | 100**1**1 | 1001**1** |
|---|---|---|---|---|---|
| X=1 | X=3 | $X=3^2$ | $X=3^4$ | $X=3^9$ | $X=3^{19}$ |

# Attacking RSA Implementations




- Something "extra" happens when we encounter a 1
  - We always square but we only multiply if there is a 1
  - A multiplication indicates a 1 in the secret exponent *d*
- Information about the secret exponent *d* can leak
  - *e.g.* as a variation in power consumption

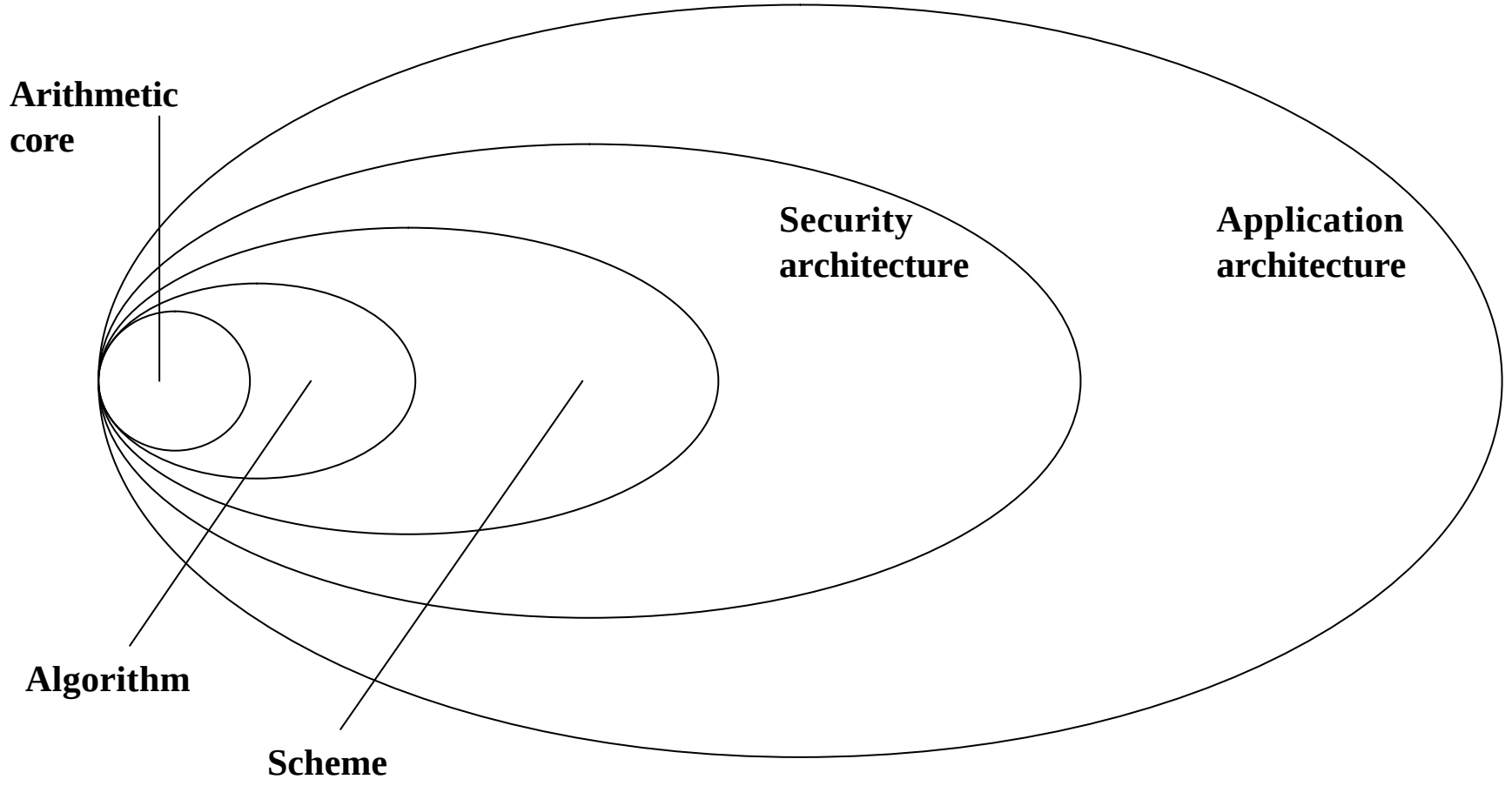
# Side-Channel Cryptanalysis

- It is very hard to make immune implementations
  - Particularly if there are restricted resources
- A new breed of attacks exploit time variation arising from using look-up tables
  - Performance is often optimised by pre-computing tables
  - The AES appears to be particularly vulnerable
- Some other attacks try to force errors during a cryptographic computation
  - The effectiveness of the attack depends on the algorithm

# Cryptography in Perspective

- When choosing an algorithm we are typically concerned with security and performance
- Not "Is this the strongest?" but "Is this strong enough?"
- Not "Is this the fastest?" but "Is this fast enough?"
- Standardized algorithms typically offer the best choice
  - Interoperability and trust

# Implementing Cryptography



# Cryptography in Perspective

- It is rare for cryptography to be broken in the field
  - One prominent exception being 802.11b wireless-LAN!
- There is so much else that can go wrong
  - There are much easier places for an attacker to look!
- Security is much more than cryptography
- Security as a whole is exceptionally hard
  - Key life-cycles, firewalls, passwords, certificates, access management, policies, rights, *etc., …*

# To Close: Three Research Directions

# Future Trends (1)

- Improved mathematical foundations
- There is an emphasis on "provable security" in the research community
- Sometimes this can be a bit controversial
  - Perhaps because too many are broken!
- Perhaps better viewed as reducible security under some assumptions
  - It does however give some guidance on how to use primitives

# Future Trends (2)

- Improved cryptanalysis
- There are always continual improvements
  - Attacks only get better
  - New approaches are still being explored, *e.g.* algebraic cryptanalysis
  - Also new stream ciphers and hash functions will appear
- Quantum computing would be particularly disruptive
  - Successful quantum computing would render many asymmetric cryptography techniques useless

(Not to be confused with quantum cryptography!)

# Future Trends (3)

- New application areas
- One significant area is "low-cost" cryptography
  - Asymmetric cryptography requires significant resources
  - Even symmetric cryptography can require many thousands of gates in hardware
- Devices like RFID tags introduce significant issues
  - There are challenging problems of security and privacy
  - Conventional cryptography doesn't fit
  - This is currently a particularly active research area

# Some Resources

- Historical
  - *The Codebreakers* – Kahn
  - *Crypto* – Levy
- Introductory
  - *Applied Cryptography (2nd edition)* – Schneier
  - *Cryptography: A Short Introduction* – Murphy and Piper
- More Technical
  - *The Handbook of Applied Cryptography* – Menezes, Oorschot, Vanstone
- Very Technical
  - Proceedings of Crypto, Eurocrypt, AsiaCrypt, FSE, and PKC

# Conclusions

- Cryptography is a very rich topic
  - It is a subject that bridges many areas of expertise
- Cryptographic deployment is often unseen
- It is exceptional that cryptography is the weakest link in a deployment
  - But we have to do it correctly!
- Cryptography is not all governments and big industry
  - Cryptography can be very helpful in protect the rights and privacy of the individual