Mobile Application Development
Junior infants crypto maths

Waterford Institute of Technology

November 19, 2016

John Fitzgerald

Sign your app

Learning objectives

An overview of:

- Mathematics underlying encryption.
- Extremely simple explanations.
- Real-world encryption uses huge numbers:
- Example: 600 digits; 2000 bits.
- We work with smallest possible quantities for learning purpose.
- We provide examples of:
  - Prime numbers.
  - Generators.
  - Modular arithmetic.
  - Symmetric key encryption.
  - Public key encryption.
  - Hashing

Number Theory

The briefest of introductions

Number Theory

Crypto maths

Prime number

- Natural numbers: whole numbers: $0, 1, 2, 3, \ldots$
- Prime: natural number divisible only by itself and one.
- Examples of primes: $2, 3, 5, 7, 11, 13$
- 4 is not prime because it is divisible by 2.
- Zero and one are not considered primes.

Crypto maths
Prime number

- There is an infinite number of primes.
- Primes still being discovered.
- Structure of pattern of primes still unsolved.
- In real-world cryptography huge prime numbers are used.
- Typically 600 digits, approximately 2000 bits.
- We will work with very small primes.

Crypto maths

Prime number

- All natural numbers are either prime or composite numbers.
- A number not a prime number is a composite.
- Prime: 7 because factors are itself and one only.
- Composite: 8 because factors are $1, 2, 4, 8$ and so not prime.

Crypto maths

Euclid's discoveries (300 BC)

- Realized all numbers prime or composite.
- Any number repeatedly divisible until set primes arrived at.
- $15 = 3 + 3 + 3 + 3 + 3$
- $25 = 5 + 5 + 5 + 5 + 5$
- $49 = 7 + 7 + 7 + 7 + 7 + 7 + 7$

Crypto maths

Euclid Fundamental Theorem of Arithmetic

Also called *Unique Factorization Theorem* or

Unique Prime Factorization Theorem

- Every integer greater than 1 either prime or product of primes
- Example: 30 = 2 x 15 (The prime 2 added 15 times)

Crypto maths

Euclid Fundamental Theorem of Arithmetic

- $30 = 2$ x 15 (The prime 2 added 15 times)
- $30 = 3$ x 10 (The prime 3 added 10 times)
- $30 = 5$ x 6 (The prime 5 added 6 times)
- 2, 3 and 5 are the prime factors of 30.

Crypto maths

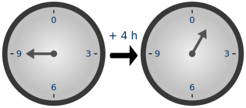
Euclid Fundamental Theorem of Arithmetic)

- 2 x 3 x 5 is prime factorization of 30.
- Every number has one & only one prime factorization.
- Unique: no two numbers have same factorization.
- Analogy: each number different lock with unique key.
- The unique key: the prime factors.
- No two locks share same key.
- No two numbers share prime same factorization.

Crypto maths

Modular arithmetic

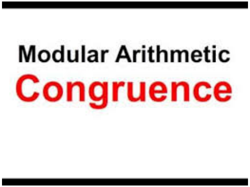
- Also referred to as *clock* arithmetic.
- Number wraps around when modulus reached.
- In case of 12-hour clock the modulus is 12
- Valid range numbers is 0 to 11.
- Example modular addition:
  - $9 + 2 \bmod 12 = 11$
  - $9 + 3 \bmod 12 = 0$
  - $9 + 4 \bmod 12 = 1$



Crypto maths

Modular arithmetic

- Java uses % operator for modular arithmetic.
- Example where modulus is 12:
  - 15 % 12 is 3
  - 3 is the remainder when 15 divided by 12.
  - Also expressed as 15 mod 12
- So 15 mod 12 is congruent to 3.
- Which may be expressed as 15 mod 12 $\equiv$ 3.



Hashing

What are hashes & how are they generated?

Hash & Hash Algorithms

Hashing

What are hashes & how are they generated?

- Cryptographic hash function:
  - Input: message variable length.
  - Output: fixed-size alphanumeric string.
  - Complexity: algorithmic complexity high.

Hashing

What are hashes & how are they generated?

- Cryptographic hash function:
  - Example: SHA-1
  - Output: fixed-size alphanumeric string.

Hashing

Trivial example hash function - definitely not cryptographic standard

```java
// Input: any-length string
// Output: 3-digit integer
static int modulus = 1000;
public static int simpleHashAlgorithm(String s) {
  int hash = 0;
  char[] chars = s.toCharArray();
  for (char ch : chars) {
    hash += ch;
  }
  return padding(hash % modulus); // padding: ensure always minimum 3 digits
}
```

```
Input: "ICTSkills-2015"
Output: 950
```

```
Input: "ICTSkills-2016"
Output: 960
```

SHA-1 hashing examples

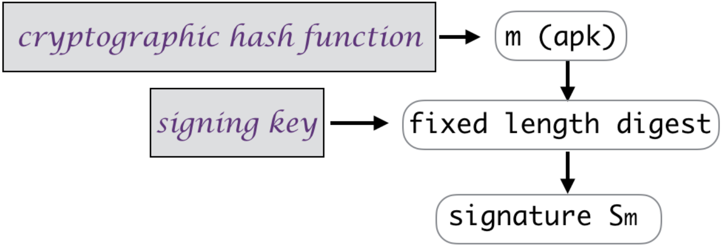
Observe differences between inputs and outputs

ICTSkills-2015
c83007996185ec1269ae9d1e78ef12d51ac0b078

ICTSkills-2016
33f87c1b7e03bc33b34e62313a638123260ca0b0

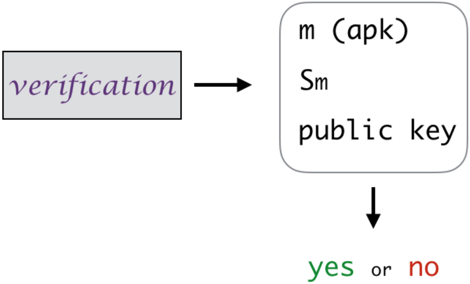
Sign your app

Signing



Sign your app

Verifying



Symmetric Key Encryption

Example using one-time pad

Symmetric Key Encryption

One Time Pad

Key same length as plaintext

Exclusive OR denoted by $\oplus$.

- m denotes plaintext or message text
- k denotes key
- c denotes the cipher text or encrypted message
- $c = m \oplus k$

a	b	$a \oplus b$
0	0	0
0	1	1
1	0	1
1	1	0

m	0	1	1	0	1	1
k	1	0	1	1	0	0
c	1	1	0	1	1	1

One Time Pad

Key same length as plaintext

Observe from table:

- $c = m \oplus k$
- $m = c \oplus k$

m	0	1	1	0	1	1
k	1	0	1	1	0	0
c	1	1	0	1	1	1
$c \oplus k$	0	1	1	0	1	1

One Time Pad

Why use XOR?

- Avoids leakage input data
- If either random variables R1 or R2 uniform
- Then R1 $\oplus$ R2 output uniform distribution
- Overcomes problem biased inputs

One Time Pad

Encrypting with logical AND

m	1 0 0 1 1	plaintext
k	1 0 1 1 0	key
m && k	1 0 0 1 0	ciphertext

Given only algorithm (&&) and ciphertext we now know that message text is
1 ? ? 1 ?

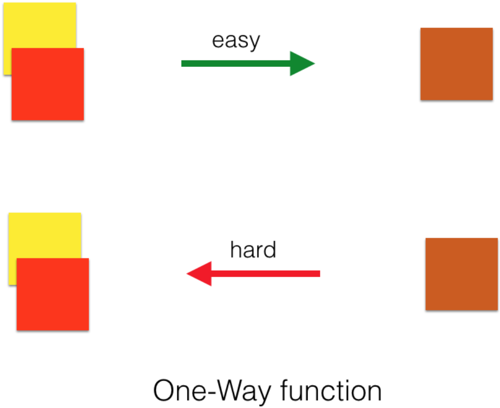
Key Exchange

Discovered independently by Diffie & Hellman (Stanford) & Malcolm Williamson (GCHQ)

Diffie-Hellman

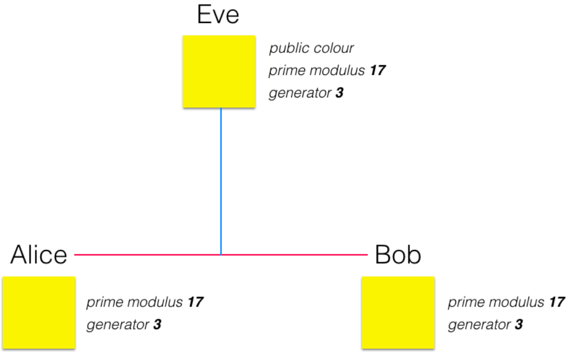
Key Exchange

Uses One-Way Function



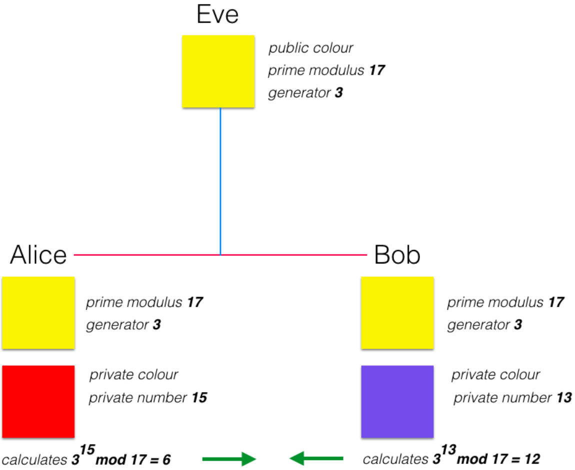
Key Exchange

One-Way Function - underlying mathematical theory



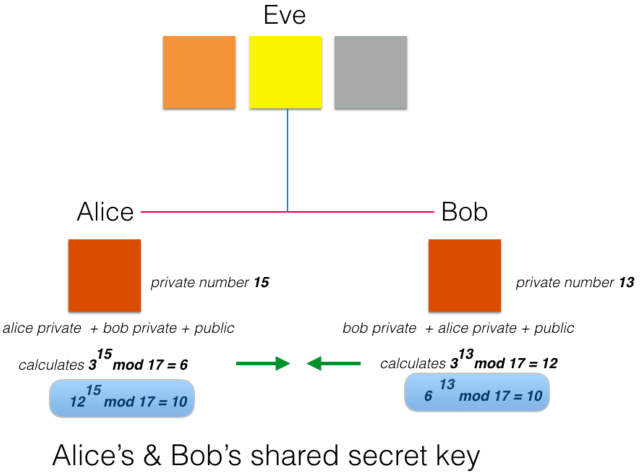
Key Exchange

One-Way Function - underlying mathematical theory



Key Exchange

One-Way Function - underlying mathematical theory



Key Exchange

Diffie-Hellman & Christopher Cocks

Diffie-Hellman

- **Public:** g and p
- **Private:** Alice's exponent a, Bob's exponent b

$\xrightarrow{g^a \bmod p}$

$\xleftarrow{g^b \bmod p}$

Alice, a Bob, b

- Alice computes $(g^b)^a = g^{ba} = g^{ab} \bmod p$
- Bob computes $(g^a)^b = g^{ab} \bmod p$
- Use $K = g^{ab} \bmod p$ as symmetric key

RSA Encryption

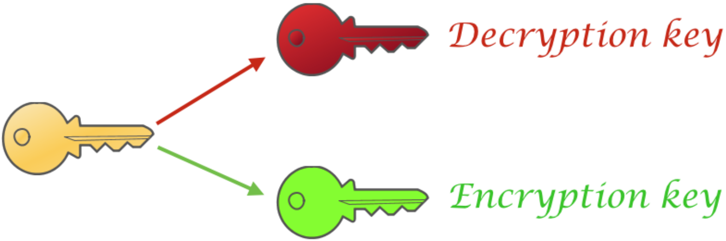
Discovered by Rivest, Shamir, Adleman (RSA) & Christopher Cocks (GCHQ)

$\mathcal{RSA}$

RSA Encryption

public-private key pair

- Ron **R**ivest, Adi **S**hamir & Leonard **A**dleman
- Key generator produces two components.
- The private (secret) key (SK) used to decrypt.
- The public key (PK) used to encrypt.
- Keys have inverse functionality.
  - Encrypt with PK => decrypt with SK.
  - Sign (encrypt) with SK => verify (decrypt) with PK.



RSA Encryption

Mathematical explanation

- Let modulus be 14.
- Alice uses key generator to output public-private key pair.
- Gives (somehow) public key to Bob.
    - Private key: (11, 14)
    - Public key: (5, 14)
    - There is a mathematical relationship between the 11 & 5
    - Brief explanation follows
    - More detailed explanations referenced materials

RSA Encryption

Mathematical explanation

- Let modulus Z be p * q where p, q very large primes
- p & q remain secret - trapdoor function
- Calculation Z very easy
- Derivation p, q given Z very hard
- Applying set rules using p, q (see ref material):
  - Choose number e.
  - (e, Z) the public key
  - Choose secret number d
  - (d, Z) the private key.

RSA Encryption

Plaintext m: Ciphertext c

- Encryption:

$$c = m^e \quad mod \quad Z$$

- Decryption:

$$m = c^d \quad mod \quad Z$$

RSA Encryption

Mathematical explanation

Bob encrypts plaintext 2 using public key (5, 14):

$$
\begin{aligned}
c &= 2^5 mod\, 14 \\
&= 4
\end{aligned}
$$

Hint: Use Paul Trow's online modular arithmetic calculator:

https://goo.gl/MhfqcO

RSA Encryption

Mathematical explanation

Alice uses private key (11, 14) to decrypt $c = 4$:

$$\begin{aligned} m &= 4^{11} mode\, 14 \\ &= 2 \end{aligned}$$

RSA Encryption

Mathematical explanation

Alice uses private key (11, 14) to sign (encrypt) a message $m = 2$:

$$\begin{aligned} c &= 2^{11} \mathit{mode}\, 14 \\ &= 4 \end{aligned}$$

Bob verifies signed message 4 using public key (5, 14):

$$\begin{aligned} m &= 4^{5} \mathit{mod}\, 14 \\ &= 2 \;\; (\mathit{verified}) \end{aligned}$$

RSA Encryption

The importance of **p** & **q**

The modulus **Z** is the product of **p** & **q**.

- Individual numbers **p** & **q** required in calculation of:
  - **e** the public key exponent (e, Z)
  - **d** the private key (d, Z)
- **Z** is public.
- Therefore: if Z factorizable then boom goes eCommerce and lots more besides.

References

Encryption & Digital Signing

1. Official documentation: Sign Your App

http://bit.ly/2eIDwQE [Accessed 2016-10-19]

2. Khan Academy: Journey into Cryptography

http://bit.ly/2eIyBPO

[Accessed 2016-10-27]

3. Mathematical Cryptosystems (1 of 2): Symmetric Cryptography

http://bit.ly/2ey52Ti

[Accessed 2016-10-27]

References

Encryption & Digital Signing

4. Mathematical Cryptosystems (2 of 2): Symmetric Cryptography

`http://bit.ly/2eOTpFV`

[Accessed 2016-10-27]

5. KhanAcademy: Digital Signatures High-level Description

`http://bit.ly/2eUCT5I`

[Accessed 2016-10-27]

6.Public Key Encrpyion & Digital Signature: How do they work?

`https://goo.gl/lHHsRo`

[Accessed 2016-10-28]

References

Encryption & Digital Signing

7. How PGP Works

https://goo.gl/2UKnR5

[Accessed 2016-10-28] 8. Diffie-Hellman key exchange

https://goo.gl/lNXGB8

[Accessed 2016-11-03]

9. How RSA and PKI works and the math behind it

http://bit.ly/2fOnvfs

[Accessed 2016-11-13]

References

Encryption & Digital Signing

10. Stackoverflow: Why is XOR used on Cryptography?

`https://goo.gl/xD37ff`

[Accessed 2016-11-15]

Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

eLearning support unit