COMx501: Computer Security and Forensics

Achim D. Brucker

a.brucker@sheffield.ac.uk **https://www.brucker.ch/**

Software Assurance & Security Research
Department of Computer Science, The University of Sheffield, Sheffield, UK
https://logicalhacking.com/

February 12, 2018

COMx501: Computer Security and Forensics

Part 5: Cryptographic Foundations

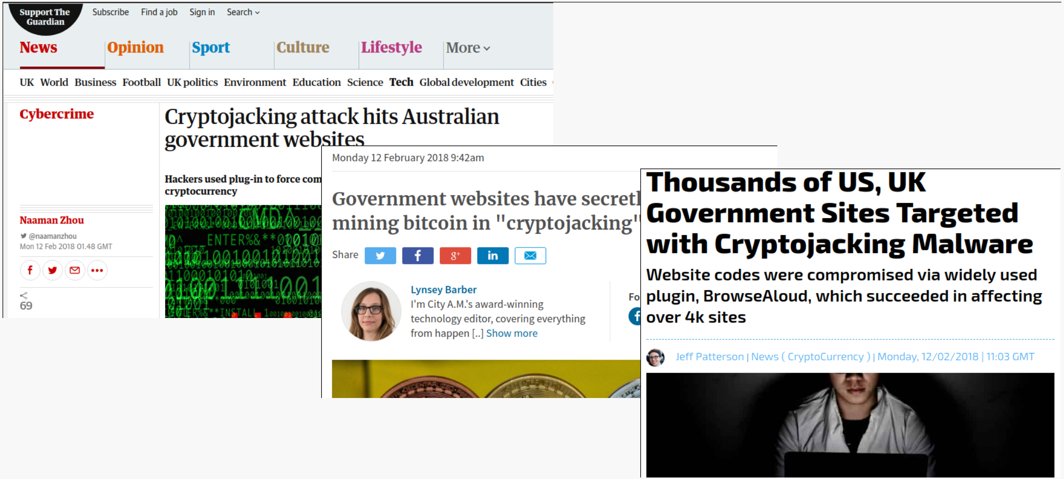
Achim D. Brucker

a.brucker@sheffield.ac.uk **https://www.brucker.ch/**

Software Assurance & Security Research
Department of Computer Science, The University of Sheffield, Sheffield, UK
https://logicalhacking.com/

February 13, 2018

Headlines from Yesterday (February 12, 2018)



Support The Guardian
Subscribe Find a job Sign in Search

News Opinion Sport Culture Lifestyle More

UK World Business Football UK politics Environment Education Science Tech Global development Cities

Cybercrime

Cryptojacking attack hits Australian government websites

Hackers used plug-in to force com[...] cryptocurrency

Naaman Zhou
@naamanzhou
Mon 12 Feb 2018 01.48 GMT

69

Monday 12 February 2018 9:42am

Government websites have secretl[...] mining bitcoin in "cryptojacking"[...]

Share

Lynsey Barber
I'm City A.M.'s award-winning technology editor, covering everything from happen [..] Show more

Thousands of US, UK Government Sites Targeted with Cryptojacking Malware

Website codes were compromised via widely used plugin, BrowseAloud, which succeeded in affecting over 4k sites

Jeff Patterson | News (CryptoCurrency) | Monday, 12/02/2018 | 11:03 GMT

What is Cryptojacking

Definition (Cryptojacking)

The mining of crypto-currencies on websites without the consent of the users.

- The attacker uses resources of the victim (computing power, i.e., costs for electricity) to gain a benefit by mining a crypto-currency
- Usually Monero (an Altcoin, i.e., an alternative to Bitcoin)
- The JavaScript code for mining is usually from coinhive.com
- Not only websites, Chrome extensions do this as well: https://logicalhacking.com/blog/2017/09/23/more-than-one-bitcoin-mining-chrome-extensions/

How to Install Crypto Mining Code on Thousands of Websites

The Root Cause: Inconsiderately Including Third Party Code

- This cryptojacking campaing mostly exploited two weaknesses:
  - A widely used JavaScript library (Browsealoud) had a vulnerability that allowed the attacker to inject the coinhive code
  - Thousands of website included the Browsealoud script without checking its integrity (hash)

Notes:

- We do not know (yet), which vulnerability enabled the attack – still, software vulnerabilities will be covered in a few weeks time
- Hashing, a method that can help to check the integrity of included files, will be discussed in this part of the lecture

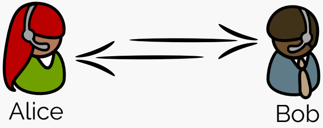
Lesson learned:

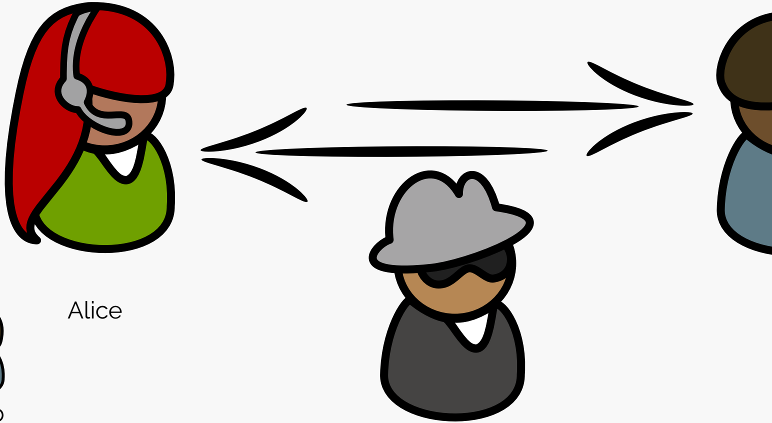
- Always be careful when including third party modules/libraries. You are responsible for their security vulnerabilities!

Outline

Motivation

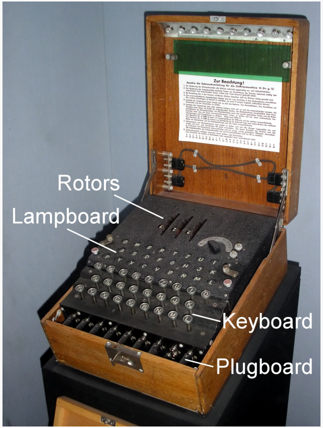
- How can we turn a untrustworthy channel into a trustworthy one?





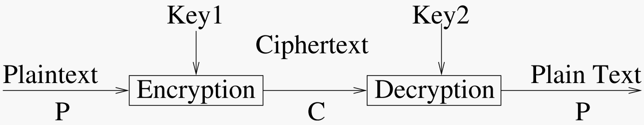
Clarifying Notation: Cryptography, Steganography, and Cryptanalysis

- Cryptography: the science of secret writing
- Steganography: the science of hiding messages in other messages or images
- Cryptanalysis: the science of analyzing a cryptographic system to break/circumvent its protection



https://en.wikipedia.org/w/index.php?title=Enigma_machine&oldid=764760662

A General Cryptographic Schema



where $E_{\mathrm{Key}_1}(P) = C$ and $D_{\mathrm{Key}_2}(C) = P$, hence: $D_{\mathrm{Key}_2}(E_{\mathrm{Key}_1}(P)) = P$

- Symmetric encryption
  - $\mathrm{Key}_1 = \mathrm{Key}_2$ (or can be easily derived from each other)
- Asymmetric encryption (public key)
  - $\mathrm{Key}_1 \neq \mathrm{Key}_2$ (cannot be easily derived from each other)
  - the public key (Key_1) can be published without compromising the private key (Key_2)
- Encryption and decryption should be easy, if keys are known.
- Security depends on secrecy of the key, not the encryption/decryption algorithms

Outline

Encryption & Decryption

We introduce:

- a finite set $\mathcal{A}$, called the alphabet.
- the message space $\mathcal{M} \subseteq \mathcal{A}^*$ and $M \in \mathcal{M}$ is a plaintext (message)
- the ciphertext space $\mathcal{C}$, whose alphabet may differ from $\mathcal{M}$
- $\mathcal{K}$ denoting the key space of keys

Moreover

- each $e \in \mathcal{K}$ determines a bijective function from $\mathcal{M}$ to $\mathcal{C}$, denoted by E_e
  E_e is the encryption function
- for each $d \in \mathcal{K}$, D_d denotes a bijection from $\mathcal{C}$ to $\mathcal{M}$
  D_d is the decryption function

Applying E_e (or D_d) is called encryption (or decryption)

Encryption (and Decryption) Schemes

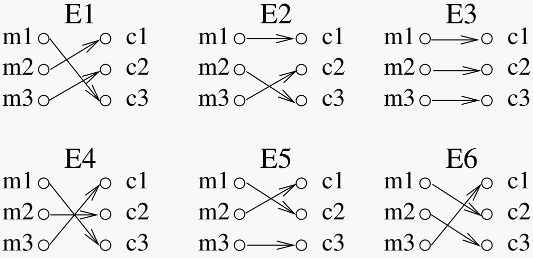
An encryption scheme (or cipher) consists of a set $\{E_e \mid e \in \mathcal{K}\}$ and a corresponding set $\{D_d \mid d \in \mathcal{K}\}$ such that for each $e \in \mathcal{K}$ there is a unique $d \in \mathcal{K}$ with $D_d = E_e^{-1}$; i.e.,

$$D_d\big(E_e(m)\big) = m, \text{ for all } m \in \mathcal{M}$$

- The keys e and d form a key pair, sometimes denoted by (e, d)
  They can be identical (i.e., the symmetric key) of a symmetric encryption scheme
- To construct an encryption scheme requires fixing a message space $\mathcal{M}$, a ciphertext space $\mathcal{C}$, and a key space $\mathcal{K}$, as well as encryption transformations $\{E_e \mid e \in \mathcal{K}\}$ and corresponding decryption transformations $\{D_d \mid d \in \mathcal{K}\}$.

An Example

- Let $\mathcal{M} = \{m_1, m_2, m_3\}$ and $\mathcal{C} = \{c_1, c_2, c_3\}$
- There are $3! = 6$ bijections from $\mathcal{M}$ to $\mathcal{C}$
- The key space $\mathcal{K} = \{E_1, E_2, E_3, E_4, E_{5,E} 6\}$ on the right
  specifies these transformations
- Assume Alice and Bob agree on E_1
- To encrypt m_1, Alice computes $E_1(m_1) = c_3$
- Bob decrypts c_3 by reversing the arrows on the diagram for E_1 and observing that c_3 points to m_1



Outline

Codes

Code

- a string of symbols stands for a complete message
- One of the simplest and earliest forms of cryptography
- Translation given by a “code-book”
- Still used today:

> “
>
> This year’s trial of the embassy bombings revealed that Bin Laden associates began to use encryption before 1998. Sometimes members of the Al-Qaida confederation have alternatively resorted to simple code words. For instance, “working” is said to mean Jihad, “tools” meant weapons, “potatoes” meant grenades and “the director” was an alias for Bin Laden.
>
> *Lisa Krieger, Mercury News, Oct 1. 2001*

Mono-Alphabetic Substitution Ciphers

- Simplest kind of cipher (idea over 2,000 years old)
- Let $\mathcal{K}$ be the set of all permutations on the alphabet $\mathcal{A}$.
  For each $e \in \mathcal{K}$, we define an encryption transformation E_e on strings $m = m_1 m_2 \cdots m_n \in \mathcal{M}$ as

$$E_e(m) = e(m_1)e(m_2)\cdots e(m_n) = c_1 c_2 \cdots c_n = c$$

- To decrypt c, compute the inverse permutation $d = e^{-1}$ and

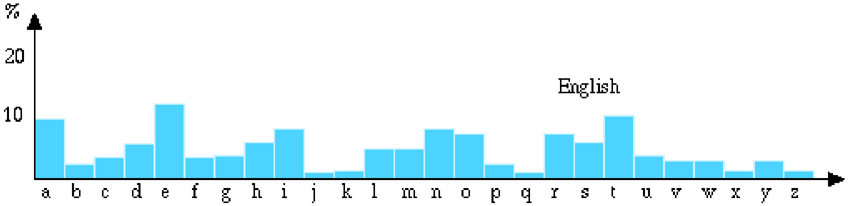
$$D_d(c) = d(c_1)d(c_2)\cdots d(c_n) = m$$

- E_e is a simple substitution cipher or a mono-alphabetic substitution cipher.

Examples of Substitution Cipher

- D(`KHOOR ZRUOG`) = `HELLO WORLD`
  - Caesar cipher:
    each plaintext character is replaced by the character three to the right modulo 26
    (e.g., E(`A`) = `D`)
- D(`Zl anzr vf Nqnz`) = `My name is Adam`
  - ROT13: shift each letter by 13 places
    On the Linux command line: `tr a-zA-Z n-za-mN-ZA-M`
- D(`2-25-5 2-25-5`) = `BYE BYE`
  - Alphanumeric: substitute numbers for letters
- How hard are these to break?

- Key spaces are typically huge. 26 letters ⇝ 26! possible keys
- Trivial to crack using frequency analysis (letters, digraphs, etc.)
- Frequencies for English based on data-mining books/articles



Serdhrapvrf sbe Ratyvfu onfrq ba qngn-zvavat obbxf/negvpyrf

- Easy to apply, except for short, atypical texts
- More sophistication required to mask statistical regularities

Polyalphabetic substitution ciphers

- Idea (Leon Alberti):
  conceal distribution using family of mappings
- A polyalphabetic substitution cipher is a block cipher with block length t over alphabet $\mathcal{A}$ where:
  - the key space $\mathcal{K}$ consists of all ordered sets of t permutations over $\mathcal{A}$, $(p_1, p_2, \dots, p_t)$
  - Encryption of $m = m_1 \cdots m_t$ under key $e = (p_1, \cdots, p_t)$ is $E_e(m) = p_1(m_1) \cdots p_t(m_t)$
  - Decryption key for e is $d = (p_1^{-1}, \cdots p_t^{-1})$

- Key given by sequence of numbers $e = e_1, \ldots, e_t$, where

$$p_i(a) = (a + e_i) \bmod n$$

  defining a permutation on an alphabet of size n
- Example: English ($n = 26$), with k = 3,7,10

$$m = \texttt{THI SCI PHE RIS CER TAI NLY NOT SEC URE}$$

  then

$$E_e(m) = \texttt{WOS VJS SOO UPC FLB WHS QSI QVD VLM XYO}$$

One-time pads (Vernam cipher)

- A one-time pad is a cipher defined over $\{0, 1\}$.
- A Message $m_1 \cdots m_n$ is encrypted by a binary key string $k_1 \cdots k_n$:

$$
\begin{aligned}
E_{k_1 \cdots k_n}(m_1 \cdots m_n) &= (m_1 \oplus k_1) \cdots (m_n \oplus k_n) \\
D_{k_1 \cdots k_n}(c_1 \cdots c_n) &= (c_1 \oplus k_1) \cdots (c_n \oplus k_n)
\end{aligned}
$$

- Example:

$$
\begin{array}{rcl}
m & = & 010111 \\
k & = & 110010 \\
\hline
c & = & 100101
\end{array}
$$

- Since every key sequence is equally likely, so is every plaintext!
- Unconditional (information theoretic) security, if key isn't reused!
- Moscow–Washington communication previously secured this way
- Problem? Securely exchanging and synchronizing long keys

Transposition Cipher

- For block length t, let $\mathcal{K}$ be the set of permutations on $\{1, \ldots, t\}$. For each $e \in \mathcal{K}$ and $m \in \mathcal{M}$

$$E_e(m) = m_{e(1)} m_{e(2)} \cdots m_{e(t)}$$

- The set of all such transformations is called a transposition cipher.
- To decrypt $c = c_1 c_2 \cdots c_t$ compute

$$D_d(c) = c_{d(1)} c_{d(2)} \cdots c_{d(t)} ,$$

  where d is inverse permutation.
- Letters are unchanged:
  - apply frequency analysis to reveal if ciphertext is a transposition
  - decrypt by exploiting frequency analysis for dipthongs, tripthongs, words, etc.

- C = Aduaenttlydhatoiekounletmtoihahvsekeeeleeyqonouv

A	n	d	i	n	t	h	e	e	n
d	t	h	e	l	o	v	e	y	o
u	t	a	k	e	i	s	e	q	u
a	l	t	o	t	h	e	l	o	v
e	y	o	u	m	a	k	e		

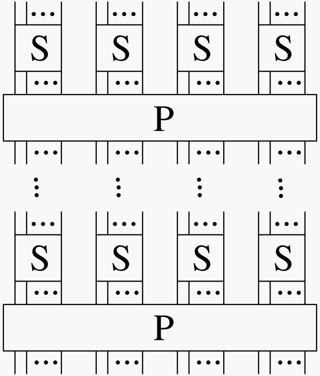
  Table defines a permutation on 1, ..., 50.

- Idea goes back to Greek Scytale:
  wrap belt spirally around baton and write plaintext lengthwise on it.

Composite Ciphers

- Ciphers based on either
  - substitutions or
  - transpositions

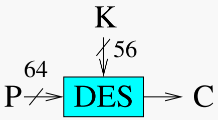
  are insecure
- Ciphers can be combined. However …
  - two substitutions are really only one more complex substitution
  - two transpositions are really only one transposition
  - but a substitution followed by a transposition makes a new harder cipher
- Product ciphers chain combinations of substitutions and transpositions
  - “S-Boxes” confuse input bits.
  - “P-Boxes” diffuse bits across S-box inputs

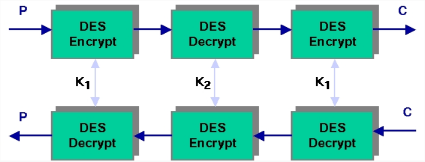


Composite Ciphers

Example: DES

- DES (Data Encryption Standard), 1993
- Block cipher, encrypting 64-bit blocks. Uses 56 bit keys
  Expressed as 64 bit numbers (8 bits parity checking)
- First cryptographic standard.
  - 1977 US federal standard (US Bureau of Standards)
  - 1981 ANSI private sector standard
- Heavily used in banking applications.
  Extensions like triple-DES used to overcome short key-length.





> People have long questioned the security of DES. There has been much speculation on the key length, number of iterations, and design of the S-boxes. The S-boxes were particularly mysterious — all those constants, without any apparent reason as to why or what they're for. Although IBM claimed that the inner workings were the result of 17 man-years of intensive cryptanalysis, some people feared that the NSA embedded a trapdoor into the algorithm so they would have an easy means of decrypting messages.
>
> Bruce Schneier, *Applied Cryptography* p278.

> The National Security Agency also provided technical advice to IBM. And Konheim has been quoted as saying "we sent the S-boxes off to Washington. They came back and were all different. We ran our tests and they passed." People have pointed to this as evidence that the NSA put a trapdoor in DES.
>
> Bruce Schneier, *Applied Cryptography* p279.

- No security proofs or reductions known
- Main attack: exhaustive search
  - 7 hours with 1 million dollar computer (in 1993).
  - 7 days with $10,000 FPGA-based machine (in 2006).
- Mathematical attacks
  - Not know yet.
  - But it is possible to reduce key space from 2^{56} to 2^{43} using (linear) cryptanalysis.
- Triple DES: use three stages of encryption
  - no known practical attack
  - brute-force search with 2^{112} operations
- DES should not be used for new applications
- “Successor” Advanced Encryption Standard (AES)

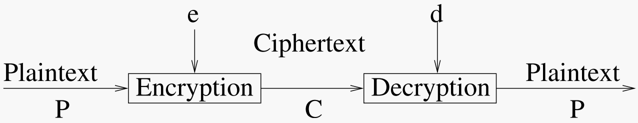
Consider an encryption scheme $\{E_e \mid e \in \mathcal{K}\}$ and $\{D_d \mid d \in \mathcal{K}\}$. The scheme is symmetric-key if for each associated pair (e, d) it is computationally "easy" to determine d knowing only e and to determine e from d. In practice $e = d$.

- Other terms: single-key, one-key, private-key, and conventional encryption.
- A block cipher is an encryption scheme that breaks up the plaintext message into strings (blocks) of a fixed length t and encrypts one block at a time.
- A stream cipher is one where the block-length is 1.
- In contrast, codes work on words of varying length.

Outline

- A function $f : X \to Y$ is a one-way function, if f is "easy" to compute for all $x \in X$, but f^{-1} is "hard" to compute.
- **Example:**
  Problem of modular cube roots
  - Select primes $p = 48611$ and $q = 53993$.
  - Let $n = pq = 2624653723$ and $X = \{1, 2, \dots, n - 1\}$.
  - Define $f : X \to N$ by $f(x) = x^3 \bmod n$.
  - Example: $f(2489991) = 1981394214$. Computing f is easy.
  - Inverting f is hard: find x which is cubed and yields remainder!
- A trapdoor one-way function is a one-way function $f : X \to Y$ where, given extra information (the trapdoor information) it is feasible to find, for $y \in Im(f)$, an $x \in X$ where $f(x) = y$.
- **Example:**
  Computing modular cube roots (above) is easy when p and q are known (basic number theory).

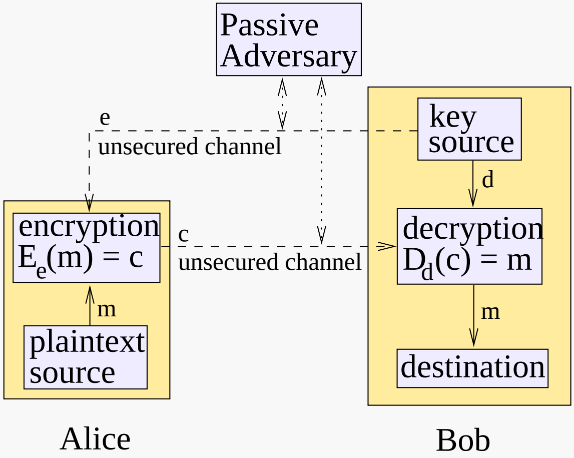
Public-Key Cryptography



- Public key cryptography is based on two keys: e and d
  - Schema designed so that given a pair (E_e, D_d),
    - knowing E_e it is infeasible,
    - given $c \in \mathcal{C}$ to find an $m \in \mathcal{M}$ where $E_e(m) = c$

    This implies it is infeasible to determine d from e
  - E_e constitutes a trap-door one-way function with trapdoor d.
- Public key e can be public information.

Encryption Using Public-Key Cryptography



When Alice can determine the message authenticity of e, public-key cryptography provides her a confidential channel to Bob

- Named after inventors: Rivest, Shamir, Adleman, 1978.
- Published after 1976 challenge by Diffie and Hellman.
- Security comes from difficulty of factoring large numbers
  Keys are functions of a pairs of large, $\geqslant 100$ digits, prime numbers
- Most popular public-key algorithm
  Used in many applications, e.g., PGP, PEM, SSL, ...
- Requires some basic number theory to appreciate.
  Warning: next five slides are technical (math)

- Numbers

$$\begin{aligned} N &= \{0, 1, 2, ...\} \\ Z &= \{0, 1, -1, ...\} \\ \text{Primes} &= \{2, 3, 5, 7, ...\} \end{aligned}$$

- Every $n \in N$ has a unique set of prime factors.
  Example: $60 = 2^2 \times 3 \times 5$
- Multiplying numbers is easy, factoring numbers appears hard.
  We cannot factor most numbers with more than 1024 bits.

- Divisors: $a \neq 0$ divides b (written $a \mid b$) if $\exists m. ma = b$
  Examples: $3 \mid 6$, $3 \;\; | 7$, $3 \;\; | 10$
- $\forall a, n. \exists q, r. a = q \times n + r$ where $0 \leqslant r < n$
  Here r is the remainder, and we write $a \bmod n = r$
- **Examples**:

$$
\begin{array}{ll}
6 = 2 \times 3 + 0 & 6 \bmod 3 = 0 \\
7 = 2 \times 3 + 1 & 7 \bmod 3 = 1 \\
10 = 3 \times 3 + 1 & 10 \bmod 3 = 1
\end{array}
$$

- $a, b \in Z$ are congruent modulo n, if $a \bmod n = b \bmod n$
  We write this as $a \equiv b \pmod{n}$.
  Example: $7 \equiv 10 \pmod{3}$

- For $a, b \in N$, $gcd(a, b)$ denotes greatest common divisor.
  Example: $60 = 2^2 \times 3 \times 5,\ 14 = 2 \times 7,\ gcd(60, 14) = 2.$
- $a, b \in N$ are relatively prime if $gcd(a, b) = 1$.
- gcd can be computed quickly using Euclid's algorithm.

$$\begin{array}{lcccl} gcd(60, 14) & : & 60 & = & 4 \times 14 + 4 \\ gcd(14, 4) & : & 14 & = & 3 \times 4 + 2 \\ gcd(4, 2) & : & 4 & = & 2 \times 2 \end{array}$$

With extended version can compute $x, y \in Z$ where

$$gcd(a, b) = xa + yb$$

Here $2 = 14 - 3 \times 4 = 14 - 3(60 - 4 \times 14) = -3 \times 60 + 13 \times 14$

- Suppose that $a, b \in Z$ are relatively prime. There is a $c \in Z$ satisfying $bc \bmod a = 1$, i.e., we can compute $b^{-1} \bmod a$.
  Proof: From extended Euclidean Algorithm, exists $x, y \in Z$ where

  $$1 = ax + by\,.$$

  Now consider the two sides modulo a. Since $a|ax$, we have $by \bmod a = 1$.
  Assertion follows with $c := y$.
- **Example:** $4^{-1} \bmod 7$
  - From Euclidean Algorithm: $1 = 7 \times (-1) + 4 \times 2$
  - Hence solution c is 2
  - Check: $4 \times 2 \bmod 7 = 1$

RSA algorithms

- Generate a public/private key pair:
  1. Generate two large distinct primes p and q.
  2. Compute $n = pq$ and $\phi = (p-1)(q-1)$.
  3. Select an e, $1 < e < \phi$, relatively prime to ϕ.
  4. Compute the unique integer d, $1 < d < \phi$ where $ed \bmod \phi = 1$.
  5. Return public key (n, e) and private key d.
- Encryption with key (n, e).
  1. Represent the message as an integer $m \in \{0, \ldots, n-1\}$.
  2. Compute $c = m^e \bmod n$.
- Decryption with key d: compute $m = c^d \bmod n$.

An RSA example

- Let $p = 47$, $q = 71$, then $n = pq = 3337$.
- Encryption key e must have no factors in common with

$$(p-1)(q-1) = 46 * 70 = 3220 \,.$$

- Choose $e = 79$ (randomly).
- Compute $d = 79^{-1} \bmod 3220 = 1019$.
- Publish e and n, keep d secret, discard p and q.
- Break message m into small blocks, e.g., $m = 688\ 232\ 687\ 966\ 668$.
- Compute $m^e \bmod n$ blockwise. E.g., $c_1 = 688^{79} \bmod 3337 = 1570$.
- To decrypt: $m_1 = 1570^{1019} \bmod 3337 = 688$.

RSA Security

- Computation of secret key d given (n, e)
  - As difficult as factorization. If we can factor $n = pq$ then we can compute $\phi = (p-1)(q-1)$ and hence $d \equiv e^{-1} \bmod \phi$.
  - No known polynomial time algorithm.
    But given progress in factoring, n should have at least 1024 bits.
- Computation of m, given c, and (n, e)
  - Computation of e-th root.
  - Unclear (= no proof) whether it is necessary to compute d, i.e., to factorize n.
- Progress in number theory could make RSA insecure.

What does this say about our modern IT (security) infrastructures?

Note on actual implementations:

- usually symmetric encryption is computational less complex (i.e., faster)
- real-world systems often based on
  - asymmetric keys-pair as long-term key
  - symmetric session key

  long-term key is used for encrypting session key

Recommendations: (https://www.keylength.com/en/4/):

- Symmetric
  - 56 bits are crackable by brute force, e.g., against DES.
  - NIST: Triple DES (with 112) and AES with at least 128 bits considered secure until 2013
  - BSI: AES with at least 128 bits considered secure until 2021
  - Usually 256 AES recommended
- Asymmetric
  - 1024-bit RSA keys considered equivalent to 80-bit symmetric keys
  - NIST: 2048 RSA considered secure until 2030
  - BSI: 3072 RSA considered secure until 2021
  - Elliptic-curve cryptography appears secure with shorter keys, e.g, 256-bits (assuming no relevant math breakthroughs)

Outline

Cryptographic Hashes: Requirements (1/2)

- Motivation: create a data “fingerprint”.
- A hash function $h(x)$ (in the general sense) has the properties:
  1. Compression: h maps an input x of an arbitrary bit length to an output $h(x)$ of fixed bit length n.
  2. Polynomial time computable.
- **Example** (longitudinal redundancy check):
  Given m blocks of n-bit input $b_1, \ldots, b_m$, form the n-bit checksum c from the bitwise xor of every block. I.e., (for $1 \leqslant i \leqslant n$)
  $$c_i = b_{i1} \oplus b_{i2} \oplus \ldots \oplus b_{im}$$
- Cryptographic techniques can be seen as a refinement of checksum techniques to handle an active forger.

Cryptographic Hashes: Requirements (2/2)

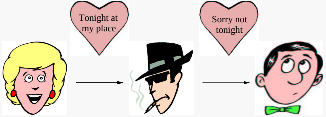
$h(x)$ is a cryptographic hash function if it is additionally:

- One-way (or pre-image resistant)
  Given y, it is hard to compute an x where $h(x) = y$.
- And usually either
  - 2nd-preimage resistance
    It is computationally infeasible to find a second input that has the same output as any specified input, i.e., given x to find an $x' \neq x$ such that $h(x) = h(x')$.
  - Collision resistance (implies 2nd-preimage resistance)
    It is difficult to find two distinct inputs x, x' where $h(x) = h(x')$.

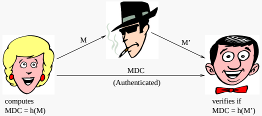
Hash value also called message digest or modification detection code (abbreviated as MDC).

Application: Message Integrity

- Message or data integrity is the property that data has not been altered in an unauthorized manner since the time it was created, transmitted, or stored by an authorized source.



- Message integrity: modification detection code provides checkable fingerprint.



Requires 2nd-preimage resistance and authenticated MDC.
Typical application: signed hashes.

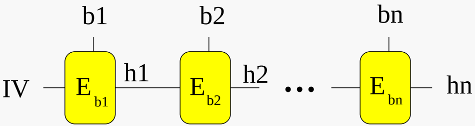
Application: Password Files

- For password p, store $h(p)$ in password file.
- Requires only pre-image resistance. Why?
- Often combined with *salt* s, i.e., store pair $(s, h(s,p))$.

Constructing a Cryptographic Hash Function

- Block chaining techniques can be used (Rabin 1978)
  - Divide message M into fixed size blocks $b_1, \dots b_n$.
  - Use symmetric encryption algorithm, e.g., DES

$$h_0 = IV \quad \text{(initial value)}$$
$$h_i = E_{b_i}(h_{i-1})$$



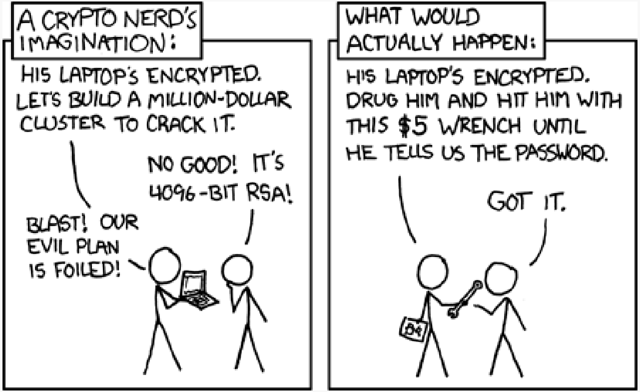
  - Similar to Cipher Block Chaining, but no secret key.
- Modern algorithms (e.g., SHA-0, MD4, MD5, ...) are much more complex and use specially designed functions.
  A number of collision results (e.g., Crypto 2004) has shaken confidence in their properties.
  Modern applications based on hashes still “appear” safe, e.g., no preimage attacks yet.

Outline

A Note on Crypto Implementations

- Implementing cryptography algorithms is a complex tasks
  - complex number theory
  - efficient implementation using machine integers (underflows, overflows, corner cases such as $|-32767| = -32767 < 0$)
- Don't implement your own crypto ...
- using existing crypto libs (e.g., OpenSSL) correctly is already a challenge
  - many algorithms, modes, and configuration options to choose from
  - complex APIs

Outlook



https://explainxkcd.com/wiki/index.php/538:_Security

- Next lectures
  - public-key infrastructures (PKI)
  - using crypto to build systems

Thank you for your attention!
Any questions or remarks?

Contact:

Dr. Achim D. Brucker
Department of Computer Science
University of Sheffield
Regent Court
211 Portobello St.
Sheffield S1 4DP, UK

a.brucker@sheffield.ac.uk
@adbrucker
https://de.linkedin.com/in/adbrucker/
https://www.brucker.ch/
https://logicalhacking.com/blog/

Bibliography I

Ross J. Anderson.
Security Engineering: A Guide to Building Dependable Distributed Systems.
John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 2001.
The complete book is available at: http://www.cl.cam.ac.uk/~rja14/book.html.

Alfred J. Menezes, Scott A. Vanstone, and Paul C. Van Oorschot.
Handbook of Applied Cryptography.
CRC Press, Inc., Boca Raton, FL, USA, 5th edition, 2001.
The complete book is available at: http://cacr.uwaterloo.ca/hac/.

Bruce Schneier.
Applied Cryptography.
John Wiley & Sons, Inc., 2 edition, 1996.

Document Classification and License Information