Modern Cryptography: Lecture 9

The Public Key Revolution I/II

Daniel Slamanig

AUSTRIAN INSTITUTE OF TECHNOLOGY

Who am I?

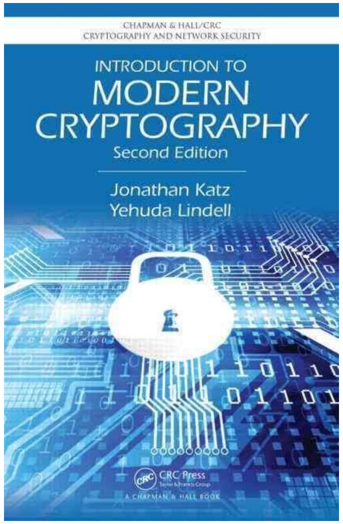
- I work as a scientist in the cryptography group at AIT in Vienna
  - Previously PostDoc and Senior Researcher at TU Graz
- AIT is Austria's largest Research and Technology Organization (RTO)
  - about 1.300 employees
- We offer internships, master and PhD student projects/supervision

Organizational

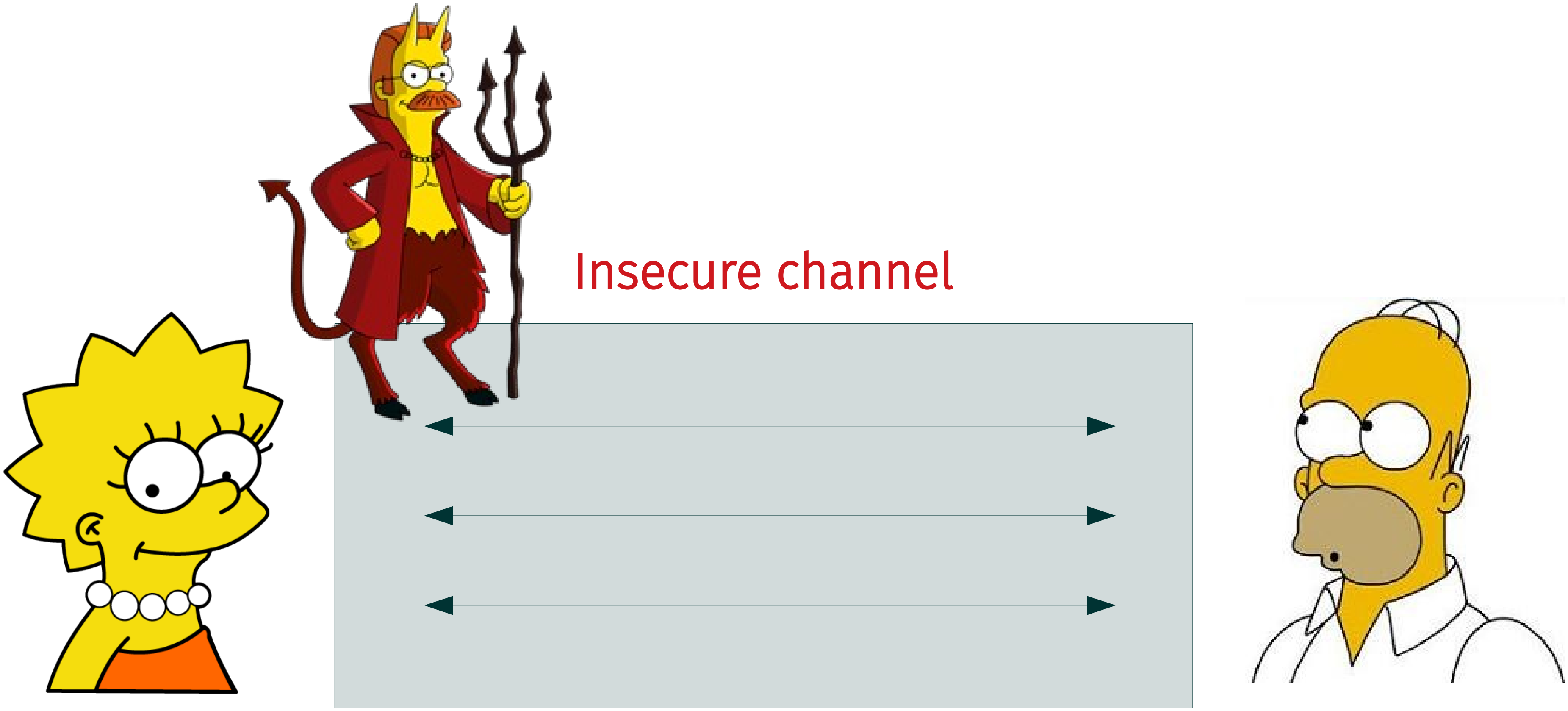
- Where to find the slides and homework?
  - https://danielslamanig.info/ModernCrypto19
- How to contact me?
  - daniel.slamanig@ait.ac.at
- Tutors: Frederick Klinser, Karen Klein
  - e11776880@student.tuwien.ac.at; karen.klein@ist.ac.at
- Official page at TU, Location etc.
  - https://tiss.tuwien.ac.at/course/courseDetails.xhtml?dswid=3463&dsrid=417&courseNr=192062&semester=2019W
- Tutorial, TU site
  - https://tiss.tuwien.ac.at/course/courseDetails.xhtml?dswid=3593&dsrid=246&courseNr=192063
- Exam for the second part: Thursday 30.01.2020 15:00-17:00 (Tutorial slot)
  - No tutorial this week → exam for first part

Outlook – Second Part

- Now we are switching to public key cryptography
- What will be covered?
  - Some basic computational number theory
  - Key exchange protocols
  - Public key encryption
  - Digital signatures
  - Selected Topics
- Invited Lecture (Dr. Christoph Striecks – AIT) – 28.01.2020
  - Advanced public key encryption (identity-based encryption and attribute-based encryption)
- The lecture next week will be held by Karen Klein



Recap: Symmetric Cryptography



Key k

Key k

Confidential & authentic message exchange

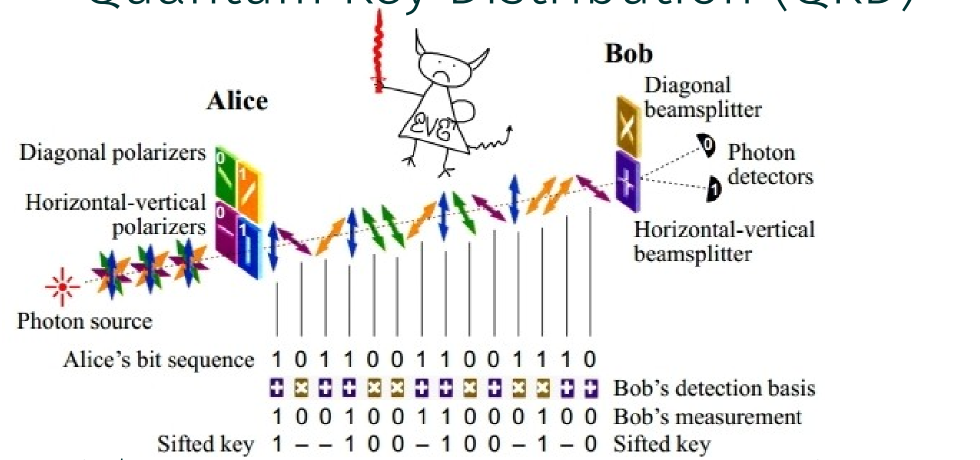
How to safely agree on the key k?

Agreeing on a common key?

- Use another channel where we can be sure there “is” **no** eavesdropper

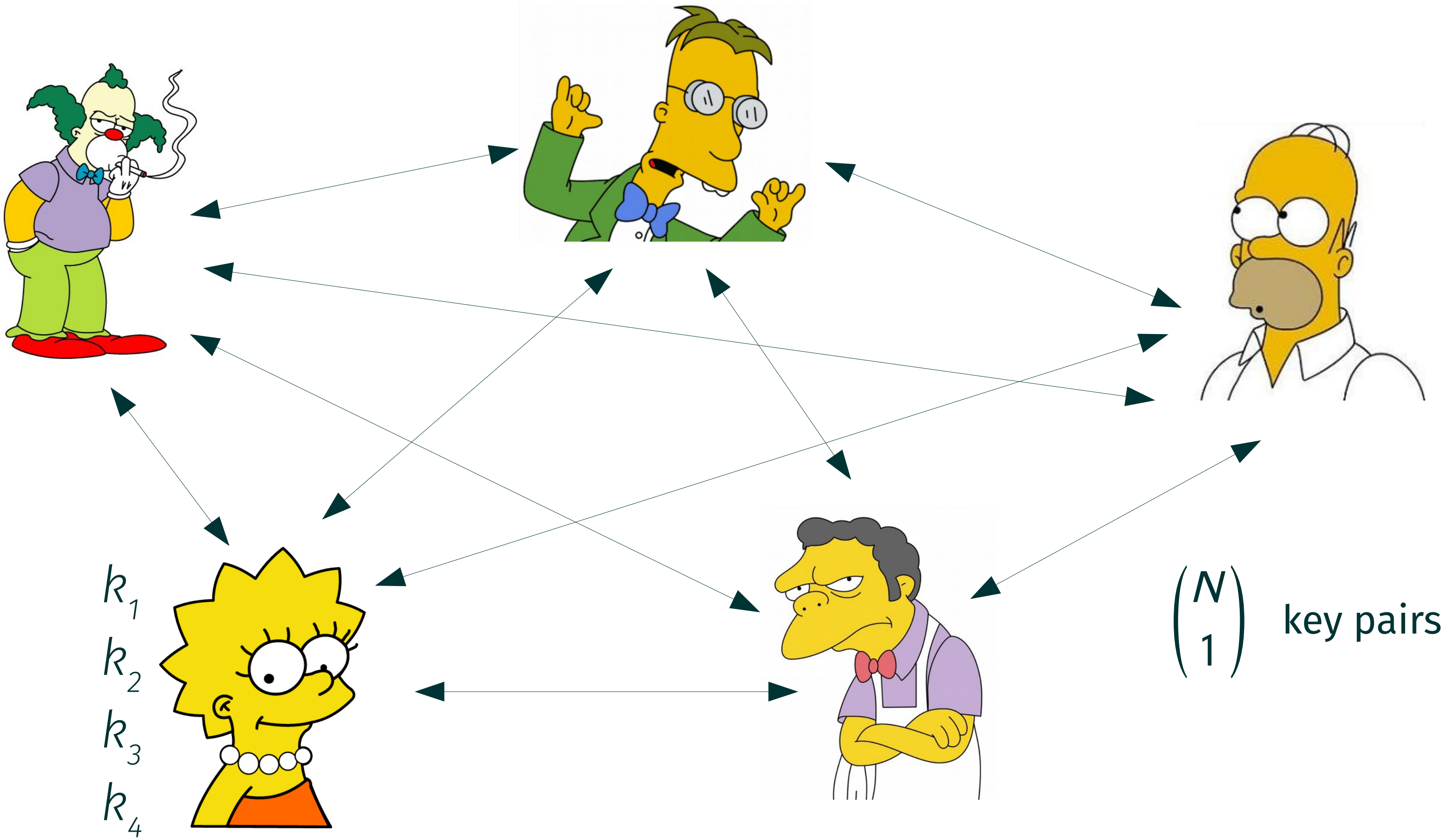
- Meeting in person?
  - “red phone” connecting Moscow and Washington in the 1960s
  - Exchange using briefcases full of prints for one-time pad encryption
- Does not “really” scale well
  - Costs, delay, ...

Quantum Key Distribution (QKD)



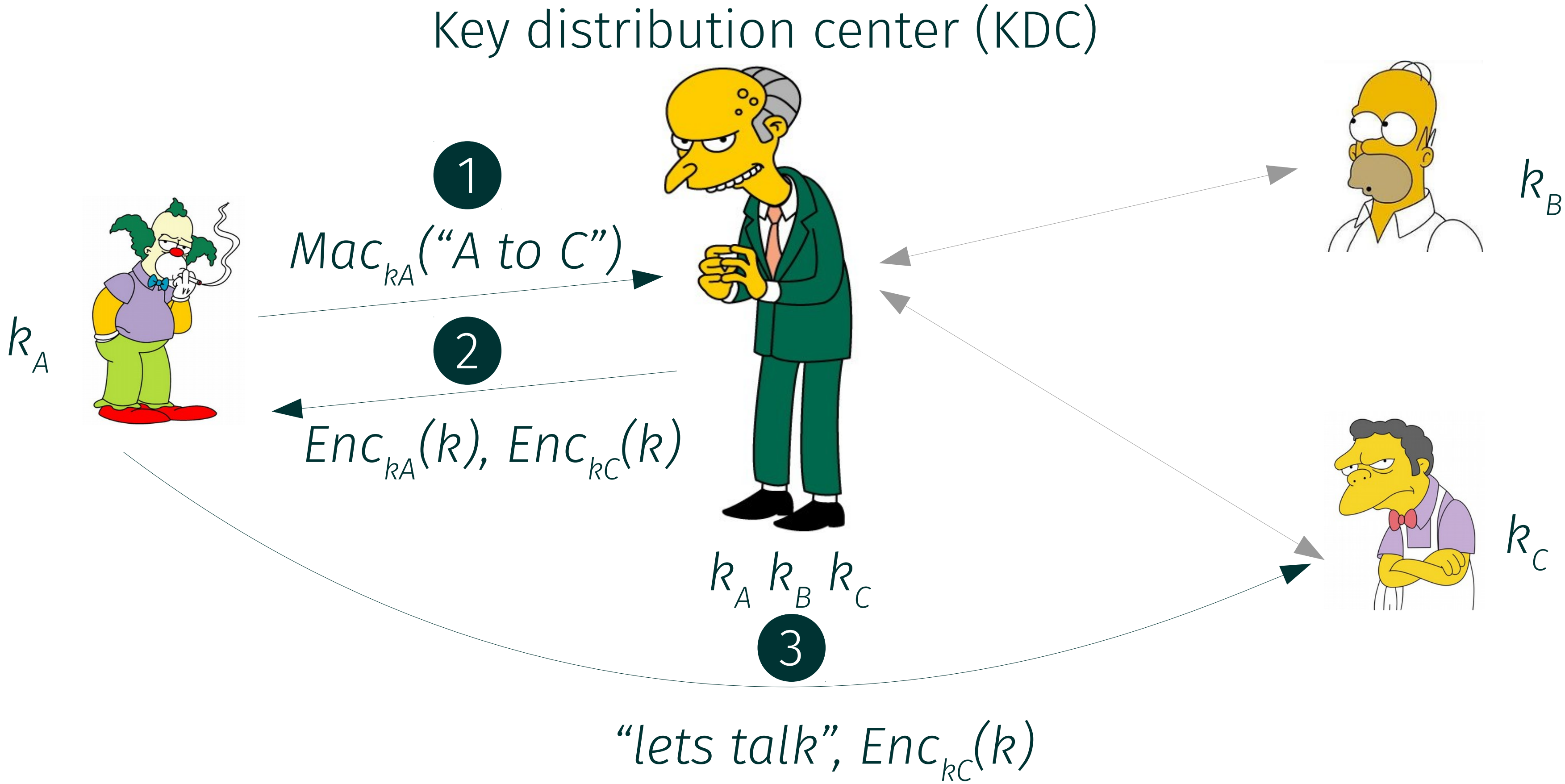
https://www.techrepublic.com/blog/it-security/how-quantum-cryptography-works-and-by-the-way-its-breakable/

Scaling to Large Networks: N^2 Problem



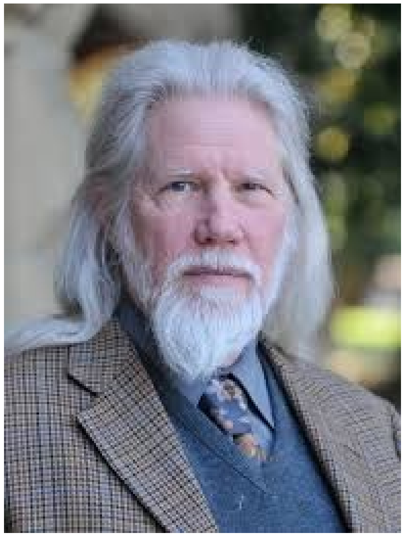
- Each of the N parties will have to store N-1 keys securely
- Cumbersone key management (update in case of loss of keys, etc.)
- Open systems?

A Partial Solution – Key Distribution Center (KDC)



- Add a trusted party (KDC) which shares a key with each party (N keys instead of N^2)
- Key updates easier, but not scalable to open systems; single point of attack
- Commonly used in <u>closed</u> systems (Kerberos, etc.)

Whitfield Diffie



Martin Hellman

Ralph Merkle

644 IEEE TRANSACTIONS ON INFORMATION THEORY, VOL. IT-22, NO. 6, NOVEMBER 1976

New Directions in Cryptography

Invited Paper

WHITFIELD DIFFIE AND MARTIN E. HELLMAN, MEMBER, IEEE

COMMUNICATIONS OF THE ACM *A Publication of the Association for Computing Machinery*

1133 AVENUE OF THE AMERICAS NEW YORK, NEW YORK 10036 212 265-6300

R. L. ASHENHURST, Editor-in-Chief
MYRTLE R. KELLINGTON, Executive Editor

Reply to:

Susan L. Graham
Computer Science Division - EECS
University of California, Berkeley
Berkeley, Ca. 94720

October 22, 1975

Mr. Ralph C. Merkle
2441 Haste St., #19
Berkeley, Ca. 94704

Dear Ralph:

Enclosed is a referee report by an experienced cryptography expert on your manuscript "Secure Communications over Insecure Channels." On the basis of this report I am unable to publish the manuscript in its present form in the Communications of the ACM.

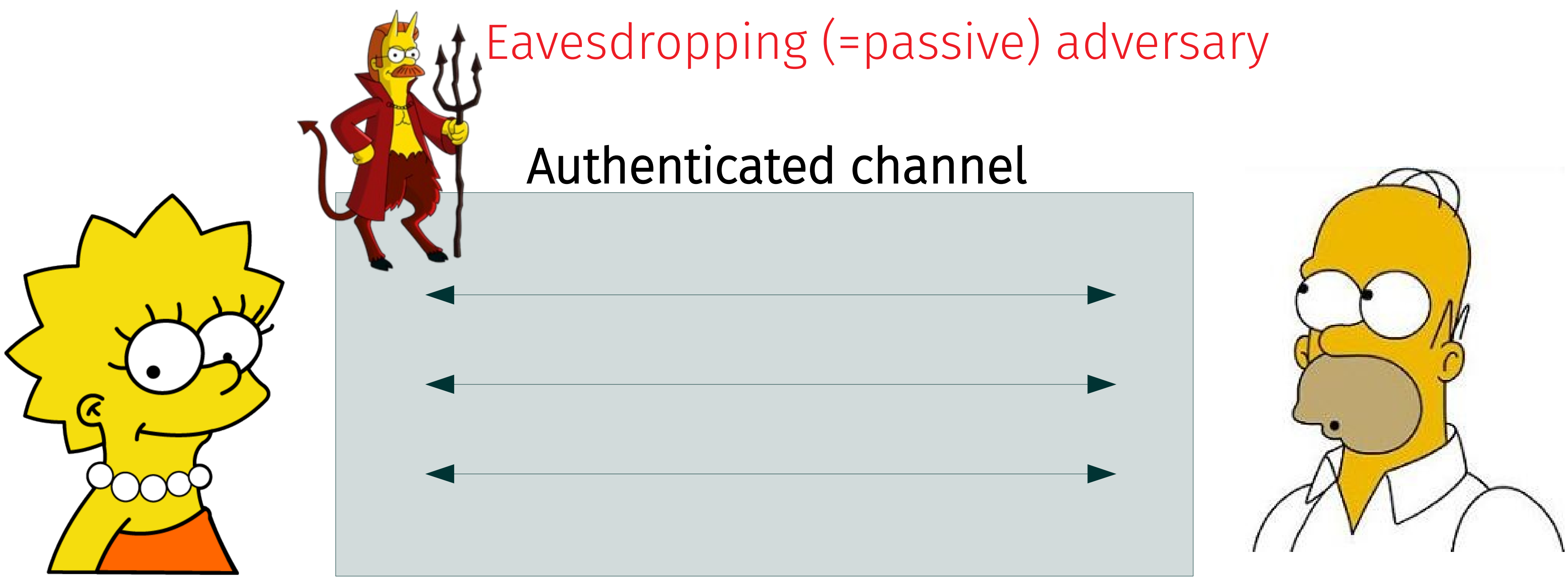
Diffie & Hellman won ACM A.M. Turing Award 2015* for fundamental contributions to modern cryptography

* "Nobel Prize of computing"

Some guys from the British signals intelligence agency (GCHQ) were even faster!

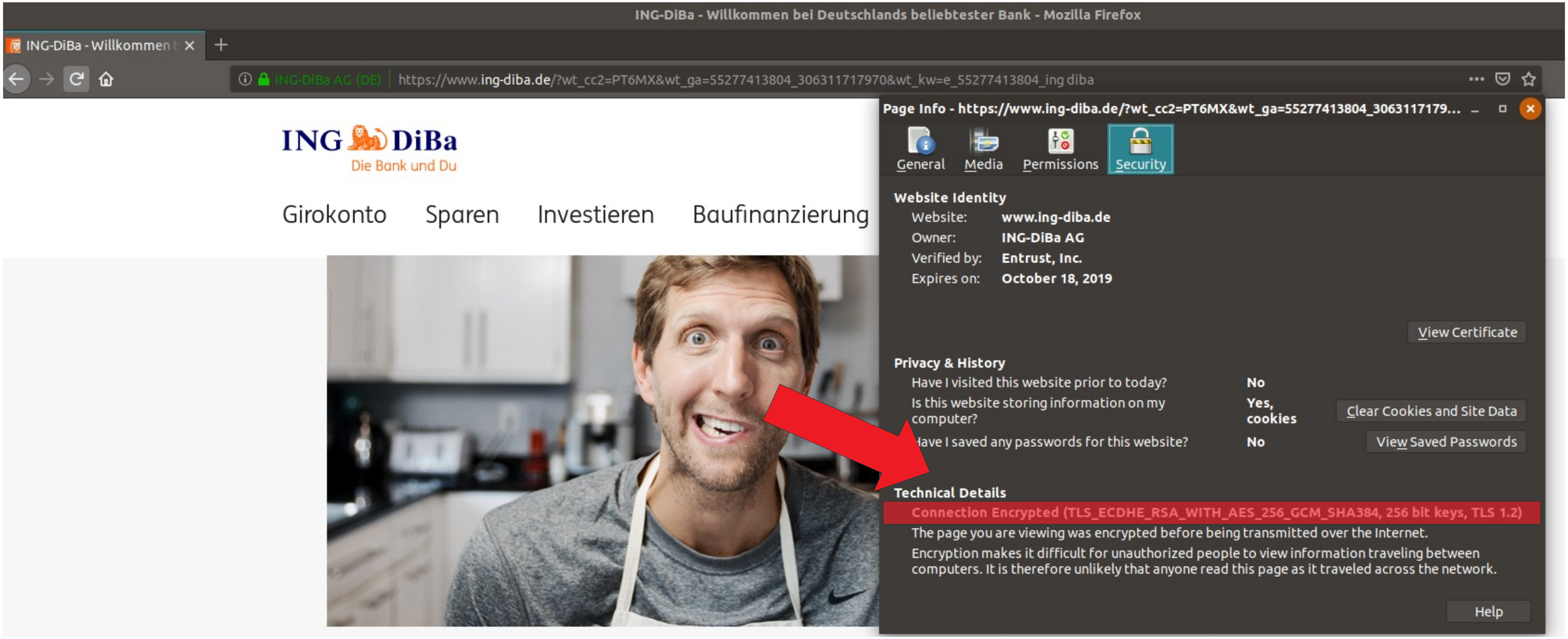
Key Exchange over Insecure Channels

- Achieve private communication <u>without</u> ever communicating over a private channel (e.g., meet personally to exchange keys)!
- Use of asymmetry in certain actions: actions that are easy to compute in one direction, but not easily reversed (one-way)
- We discuss secure key-exchange protocols à la **Diffie-Hellman** (or **Diffie–Hellman–Merkle** to be fair)



Key agreement (no prior secrets); confidential message exchange

Key Exchange – Practical Relevance



Authenticate channel using RSA signatures (PKCS#1 v1.5)

Protocol version

Connection Encrypted (TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384, 256 bit keys

TLS 1.3

Key exchange using elliptic curve DH

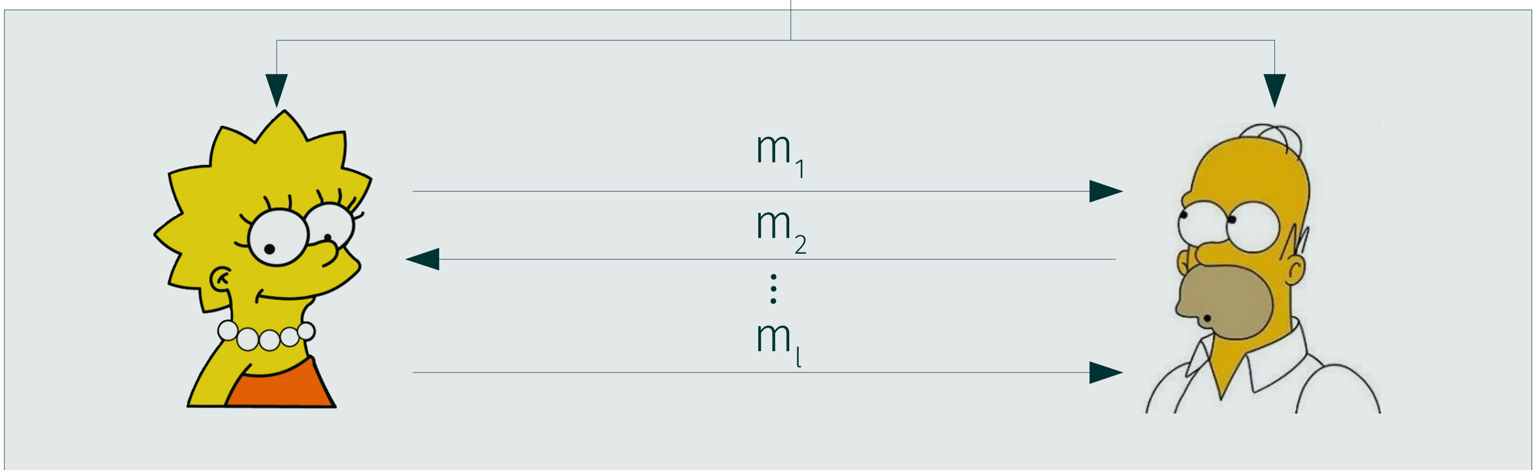
AES-256 in Galois/Counter Mode and SHA-384 as hash algorithm in HMAC

Key Exchange - Setting

- Let us consider a two-party key-exchange (KE) protocol Π

1^n **security parameter** $n \in \mathbb{N}$

Π



trans :=(m_1,m_2,...m_l)

Transcript of conversation

n-bit key k

Exchanged secret key

$\mathrm{KE}^{\mathrm{eav}}_{\mathcal{A},\Pi}$ Security §10.3

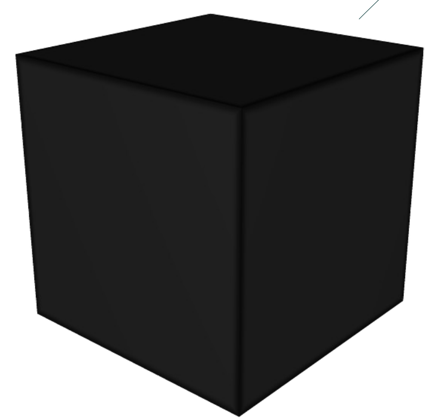
security parameter $n \in \mathbb{N}$

$\mathcal{A}$

$(k^*,\ \text{trans})$

b^*

Π



$(k,\ \text{trans})$

$b \xleftarrow{\$} \{0,1\}$

if $b = 0$

$k^* := k$

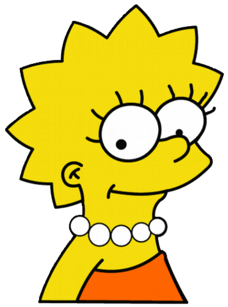
else $k^* \xleftarrow{\$} \{0,1\}^n$

A key-exchange protocol Π is secure in the presence of an eavesdropper if for every PPT adversary **A**

$$\Pr[b = b^*] \le ½ + \mathrm{negl}(n)$$

Abstract Diffie–Hellman(–Merkle) KE Protocol

$$a \xleftarrow{\$} \ldots; A \leftarrow \ldots \qquad\qquad b \xleftarrow{\$} \ldots; B \leftarrow \ldots$$



$$\xrightarrow{\quad A \quad}$$

$$\xleftarrow{\quad B \quad}$$

$$K_A \leftarrow F_A(a, B) \qquad\qquad K_B \leftarrow F_B(b, A)$$

What do we want from such a protocol?

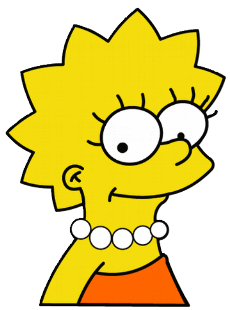
- $K_A = K_B$ so that both end up with the same shared key
- Adversary seeing A,B cannot compute K_A and K_B

How to build such a protocol?

Diffie–Hellman(–Merkle) KE Protocol

Let p be a large prime and let g be a generator mod p. Let $\mathbf{Z}_p = \{0, \ldots, p\text{-}1\}$

$$a \xleftarrow{\$} \mathbb{Z}_p; A \leftarrow g^a \mod p \qquad\qquad b \xleftarrow{\$} \mathbb{Z}_p; B \leftarrow g^b \mod p$$



$$\xrightarrow{\quad A \quad}$$

$$\xleftarrow{\quad B \quad}$$

$$K_A \leftarrow B^a \mod p \qquad\qquad K_B \leftarrow A^b \mod p$$

$B^a = (g^b)^a = g^{ab} = (g^a)^b = A^b$, so $K_A = K_B$

Adversary needs to compute g^{ab} mod p from g^a mod p and g^b mod p

How to pick p and g? How to compute g^{ab} mod p? Why is it hard for the adversary to find the shared key? How to abstract away from this concrete setting?

Some Computational Number Theory

Integers mod N

- Notation
  - $\mathbf{Z} = \{..., -2, -1, 0, 1, 2, ...\}$
  - $\mathbf{N} = \{0, 1, 2, ...\}$
  - $\mathbf{Z}_{>0} = \{1, 2, 3, ...\}$
- For $a,N \in \mathbf{Z}$ let $\mathbf{gcd}(a,N)$ be the largest $d \in \mathbf{Z}_{>0}$ s.t. $d|a$ and $d|N$
- Integers mod N. Let $N \in \mathbf{Z}_{>0}$
  - $\mathbf{Z}_N = \{0, 1, ..., N-1\}$
  - $\mathbf{Z}^*_N = \{a \in \mathbf{Z}_N : \mathbf{gcd}(a,N)=1\}$ //integers that are coprime
  - $\varphi(N) = |\mathbf{Z}^*_N|$ //number of coprime integers; $\varphi(N) = N \cdot \prod_{p|N}(1-1/p)$

Example: N=12

- $\mathbf{Z}_{12} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11\}$
- $\mathbf{Z}^*_{12} = \{1, 5, 7, 11\}$
- $\varphi(12) = 4$

Division, Remainder, Modulo

PROPOSITION 8.1 Let a be an integer and let N be a positive integer. Then there exist unique integers q, r for which $a = qN + r$ and $0 \le r < N$.

Let us write $(q,r) \leftarrow \mathbf{div}(a,N)$

- Call q the quotient and r the remainder
- Then $a \bmod N = r \in \mathbb{Z}_N$

$a \equiv b \pmod N$ if
$a \bmod N = b \bmod N$ or equivalently
$N \mid (a-b)$

Example:

- $\mathbf{div}(17,3) = (5,2)$ and $17 \bmod 3 = 2$
- $17 \equiv 14 \pmod 3$

Reduce and then add/multiply

$(a + b) \bmod N = (a \bmod N + b \bmod N) \bmod N$
$(a \cdot b) \bmod N = (a \bmod N \cdot b \bmod N) \bmod N$

Groups

- A (finite) group G is a (finite) non-empty set with a binary operation $\cdot$ s.t. the following properties hold:
  - Closure: For all $g,h \in G$, $g \cdot h \in G$
  - Identity: There exists $e \in G$ s.t. for all $g \in G$ we have $e \cdot g = g = g \cdot e$
  - Inverse: For all $g \in G$ there exists $h \in G$ s.t. $g \cdot h = e = h \cdot g$
  - Associativity: For all $g,h,f \in G$ it holds that $(g \cdot h) \cdot f = g \cdot (h \cdot f)$
- A group is commutative (or abelian) if for all $g,h \in G$ we have $g \cdot h = h \cdot g$
  - We will only deal with commutative groups

Example:

- If $N \in Z_{>0}$ then $G = Z^*_N$ with $a \cdot b \bmod N$ is a group

Exponentiation

Let us write $g^m := \underbrace{g \cdot \ldots \cdot g}_{\text{m-times}}$ for $m \in \mathbb{N}$ and $m \cdot g = \underbrace{g + \ldots + g}_{\text{m-times}}$ (for additive groups)

Also let $g^{-m} := \underbrace{g^{-1} \cdot \ldots \cdot g^{-1}}_{\text{m-times}}$

We have for all $i,j \in \mathbb{Z}$:

- $g^{i+j} = g^i \cdot g^j$
- $g^{ij} = (g^i)^j = (g^j)^i$
- $g^{-i} = (g^i)^{-1} = (g^{-1})^i$

Example: Let N=14 and $G = \mathbb{Z}^*_N$

- $5^3 = 5 \cdot 5 \cdot 5 \equiv 25 \cdot 5 \equiv 11 \cdot 5 \equiv 55 \equiv 13$

Order of a Group

Order: If G is finite, then $m := |G|$ is called the order of the group

<u>THEOREM 8.14</u> Let G be a finite group with $m = |G|$, the order of the group. Then for any element $g \in G$, $g^m = 1$.

Example: Let N=21 and $G = Z^*_N$. The order of Z^*_{21} is 12.

$5^{12} \equiv (5^3)^4 \equiv 20^4 \equiv (-1)^4 \equiv 1$

<u>COROLLARY 8.15</u> Let G be a finite group with $m = |G| > 1$. Then for any $g \in G$ and any integer x, we have $g^x = g^{[x \bmod m]}$.

Example: Let N=21 and $G = Z^*_N$. The order of Z^*_{21} is 12.

$5^{74} \equiv 5^{74 \bmod 12} \equiv 5^2 \equiv 4$

Modular Exponentiation

- For cryptographic applications we deal with very large numbers, e.g., size of exponents hundreds to thousands of bits
- How to efficiently compute a^n for large n?
- Iteratively applying group operation requires $O(n) = O(2^{|n|})$ operations: exponential time!
- Fast exponentiation idea
  - $a \to a^2 \to a^4 \to a^8 \to a^{16} \to a^{32}$
  - Use repeated squaring. If $n=2^i$ compute a^n in i steps
  - What if n is not a power of 2?

Suppose the binary length of n is 5, i.e., the binary representation of n has the form $b_4b_3b_2b_1b_0$. Then

$$n = 2^4b_4 + 2^3b_3 + 2^2b_2 + 2^1b_1 + 2^0b_0$$
$$= 16b_4 + 8b_3 + 4b_2 + 2b_1 + b_0 .$$

Computing a^n:

$$t_5 = 1$$
$$t_4 = t_5^2 \cdot a^{b4} \qquad = a^{b4}$$
$$t_3 = t_4^2 \cdot a^{b3} \qquad = a^{2b4+b3}$$
$$t_2 = t_3^2 \cdot a^{b2} \qquad = a^{4b4+2b3+b2}$$
$$t_1 = t_2^2 \cdot a^{b1} \qquad = a^{8b4+4b3+2b2+b1}$$
$$t_0 = t_1^2 \cdot a^{b0} \qquad = a^{16b4+8b3+4b2+2b1+b0}$$

Square and Multiply

- Let bin(n) := $b_{k-1}, \ldots, b_0$ with $n = \sum_{i=0}^{k-1} b_i 2^i$

ALGORITHM: Square and multiply
Input: Group element a, integer n
Output: a^n

```
b_{k-1},..., b_0 ← bin(n)
t ← 1
    for j = k-1 to 0:
        t ← t^2 · a^bi
return t
```

The algorithm requires O(|n|) group operations

Precomputations: If element a is known and there is a bound on the size of n, then one can precompute a table of powers of a. # multiplications one less than Hamming weight of bin(n).

Cyclic Groups

Let us consider a finite group G of order m and write $\langle g \rangle = \{g^0, g^1, \ldots\}$

- We know that $g^m = 1$ and now look at which elements the powers of g do "generate"
- Let $i \le m$ be the smallest positive integer for which $g^i=1$, then the above sequence repeats after i terms ($g^i=g^0$, $g^{i+1} = g^i$, …) and $\langle g \rangle = \{g^0, g^1, \ldots, g^{i-1}\}$
- We call i the order of g and $\langle g \rangle \subseteq G$ is called the subgroup generated by g
- If there is an element g with order $m:=|G|$, then G is called cyclic. We write $\langle g \rangle = G$

PROPOSITION 8.52 Let G be a finite group, and $g \in G$ an element of order i. Then for any integer x, we have $g^x = g^{[x \bmod i]}$.

PROPOSITION 8.54 Let G be a finite group of order m, and say $g \in G$ has order i. Then $i \mid m$.

Cyclic Groups - Example

Let G = Z^*_{11} = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}, which has order m = 10.

	0	1	2	3	4	5	6	7	8	9	10
2^i mod 11	1	2	4	8	5	10	9	7	3	6	1
5^i mod 11	1	5	3	4	9	1	5	3	4	9	1

‹2› = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10} and thus 2 generates Z^*_{11}

‹5› = {1, 3, 4, 5, 9} and thus 5 generates a subgroup of order 5

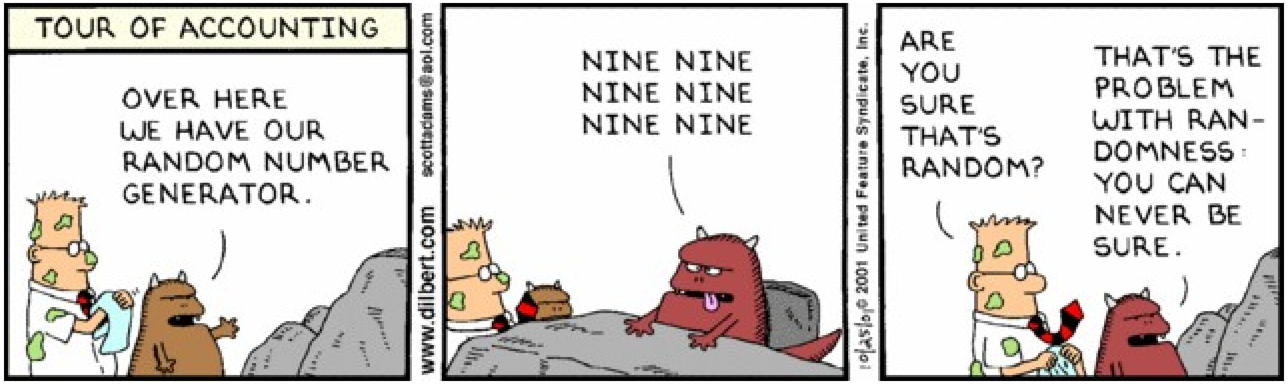
Z^*_{11} is a cyclic group (as it has a generator)

THEOREM 8.56 If p is prime then Z_p^* is a cyclic group of order p – 1.

Generating Random Primes

How to generate large random prime numbers of size used in cryptography?

5809605995369958062859502533304574370686975176362895236661486152287203730999711022573733604453311840725132615775498051744399052959454004712166288 56721
8703240103211163970644049884404985098905162720024476580704181239472968054002410482797658436938152229236120877904476989274322575173807697956881130957
912551133309324351955378481630638158016186020024749256844815024251530444957718760413642873858099017255157393414625583036640591500086964373205321856 68
3254529110790372283163413859958640669032595972518744716905954080501231020963901175074876001709536073423494575741627299485601330861695852995830467 7637
01918159408852834506128586389827176345729488354663887955431161544644633019925438234001629205709075117553388816191898729559153153669870129226768546551
7437915790823154844634780260102891718032495396075041899485513811126977307478969074857043710716150121315922024556759241239013152919710956468406379442 91
4941614357107914462567329693649



Generating Random Primes

ALGORITHM 8.31: Generating a random prime
Input: Length n; parameter t
Output: A uniform n-bit prime

```
for i = 1 to t:              // try t times
    p' ←$ {0, 1}^(n-1)       // randomly sample n-1 bits
    p := 1||p'               // n-bit integer
    if p is prime return p   //check for primality
return fail
```

How to choose t s.t. we will catch a prime with high probability?

THEOREM 8.32 (Bertrand's postulate):* For any $n > 1$, the fraction of n-bit integers that are prime is at least $1/3n$.
* the prime number theorem gives a better bound.

Setting $t=3n^2$ the probability that we do not hit any prime in t iterations is negligible.

How to implement the test "if p is prime"?

Probabilistic Primality Test

Although there are deterministic primality tests, we use probabilistic ones (as they are more efficient).

Probabilistic tests of the form: if the input n is a prime number, the algorithm always outputs "prime." If n is composite, then the algorithm would almost always output "composite," but might output the wrong answer ("prime") with a certain probability (composite is definite, for prime it can err).

COROLLARY 8.21 (Euler/Fermat): Take an arbitrary integer $N > 1$ and $a \in Z_N^*$. Then $a^{\varphi(N)} = 1 \bmod N$.
For the specific case that $N = p$ is prime and $a \in \{1, \ldots, p-1\}$, we have $a^{p-1} = 1 \bmod p$.

The Fermat test: Given n, for i=1 to t: pick $a \xleftarrow{\$} \{1, \ldots, n-1\}$ and if $a^{n-1} \neq 1 \bmod n$ output "composite". Output "prime".

Unfortunately, there are "Fermat pseudo-primes" (Carmichael numbers), which are composite but fool the test for any a.

Primality Testing in Practice

- Typically combine some pre-processing and Miller-Rabin
  - Look up in first x primes, trial divisions with first y primes, fixed-base Fermat test
  - Then run e.g., t=40 rounds of Miller-Rabin
- Some don't do a good job!

Primality testing in Apple core…crypto

Prime and Prejudice: Primality Testing Under Adversarial Conditions

Martin R. Albrecht[1], Jake Massimo[1], Kenneth G. Paterson[1], and Juraj Somorovsk…[2]

[1] Royal Holloway, University of London
[2] Ruhr University Bochum, German…
martin.albrecht@rhul.ac.uk, jake.massimo.2015@rhul … …rhul.ac.uk,
juraj.somorov…

Abstract. This … …atic analysis of primality testing under adversarial conditions, where … being tested for primality are not generated randomly, but instead provided by … possibly malicious party. Such a situation can arise in secure messaging

Craft composite numbers that will pass primality tests!

Today Apple publish their security update (https://support.apple.com/en-gb/HT201222) for macOS Mojave 10.14.1 and iOS 12.1, which includes changes to the way they test numbers for primality. In this post I will describe how easily we could produce composite numbers that fool Apple into classifying as prime what exactly has changed to the primality testing in this update.

Finding Generators: How many are there?

THEOREM B.16: Let G be a cyclic group of order $q > 1$ with generator g. There are $\varphi(q)$ generators of G, and these are exactly given by $\{g^x \mid x \in \mathbb{Z}^*_q\}$.

- Proof: Consider an element $h \neq 1$. We can write $h = g^x$ for some $1 \leq x < q$
  - If $\gcd(x,q) = r > 1$: Then $x = \alpha r$ and $q = \beta r$ with $1 \leq r < q$. Then we have $h^\beta = (g^x)^\beta = g^{\alpha r \beta} = (g^q)^\alpha = 1$. So h cannot be a generator.
  - If $\gcd(x,q) = 1$: Let $i \leq q$ be the order of h. Then $g^0 = 1 = h^i = (g^x)^i = g^{xi}$, and so $xi = 0 \bmod q$ and thus $q \mid xi$. As $\gcd(x,q) = 1$ we have $q \mid i$ and so $q = i$. Thus, h is a generator.

COROLLARY 8.55 If G is a group of prime order p, then G is cyclic. Furthermore, all elements of G except the identity are generators of G.

Finding Generators: How to find them?

PROPOSITION B.17 Let G be a group of order q, and let $q = \prod^{p}_{i=1} p_i^{e_i}$ be the prime factorization of q, where the p_i are distinct primes and $e_i \geq 1$. Set $q_i = q/p_i$. Then $h \in G$ is a generator of G if and only if $h^{q_i} \neq 1$ for i = 1, ..., k.

If we do not know the factorization of q, then we could simple enumerate trough all elements to check if an element is a generator (inefficient!).

The known factorization suggests a more efficient algorithm.

ALGORITHM B.18: Testing for generators
Input: Group order q, factors $\{p_i\}$ of q, element h
Output: A decision bit

```
for j = 1 to k:
    if h^(q/p_i) =1 return "false"
return "true"
```

Isomorphism of Cyclic Groups

<u>EXAMPLE 8.61</u>: Let G be a cyclic group of order n, and let g be a generator of G. Then the mapping $f : \mathbb{Z}_n \to G$ given by $f(a) = g^a$ is an isomorphism between $\mathbb{Z}_n$ and G. Indeed, for $a, a' \in \mathbb{Z}_n$ we have $f(a + a') = g^{[a+a' \bmod n]} = g^{a+a} = g^a \cdot g^{a'} = f(a) \cdot f(a')$.

From an algebraic point of view all cyclic groups are the “same”.

We have seen that f is easy to compute generically (square-and-multiply). However, from an computational point of view in particular f^{-1} does not need to be efficiently computable.

We will formalize this as the discrete logarithm problem.