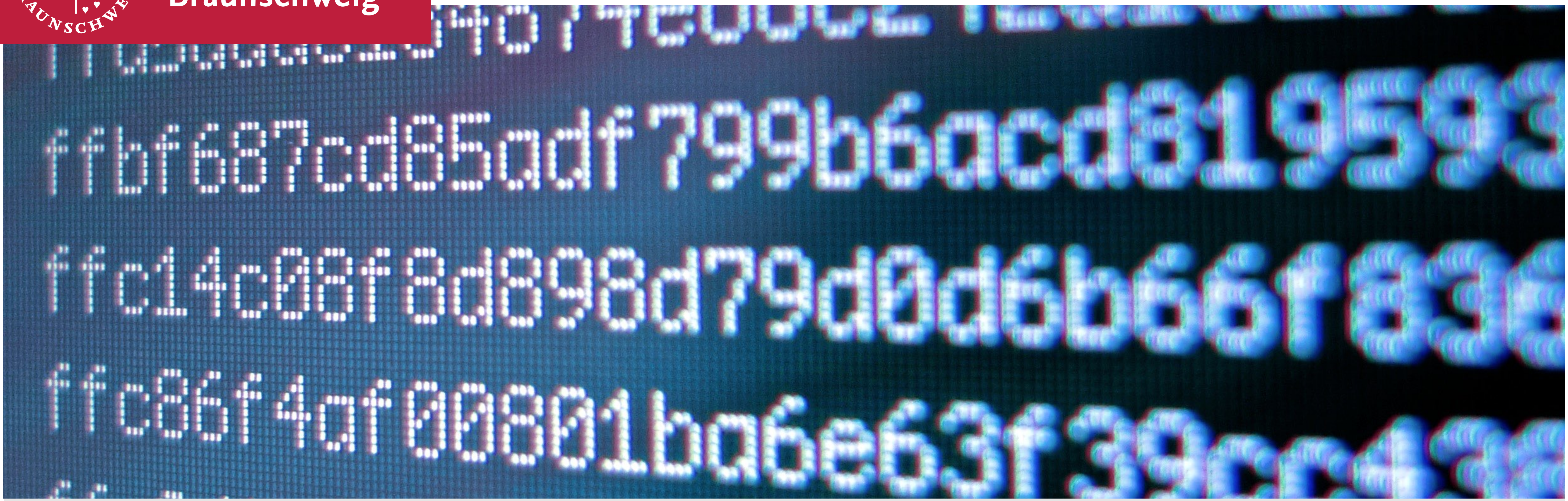
Technische Universität Braunschweig

Institute of System Security



Public-Key Cryptography

Vorlesung “Einführung in die IT-Sicherheit”

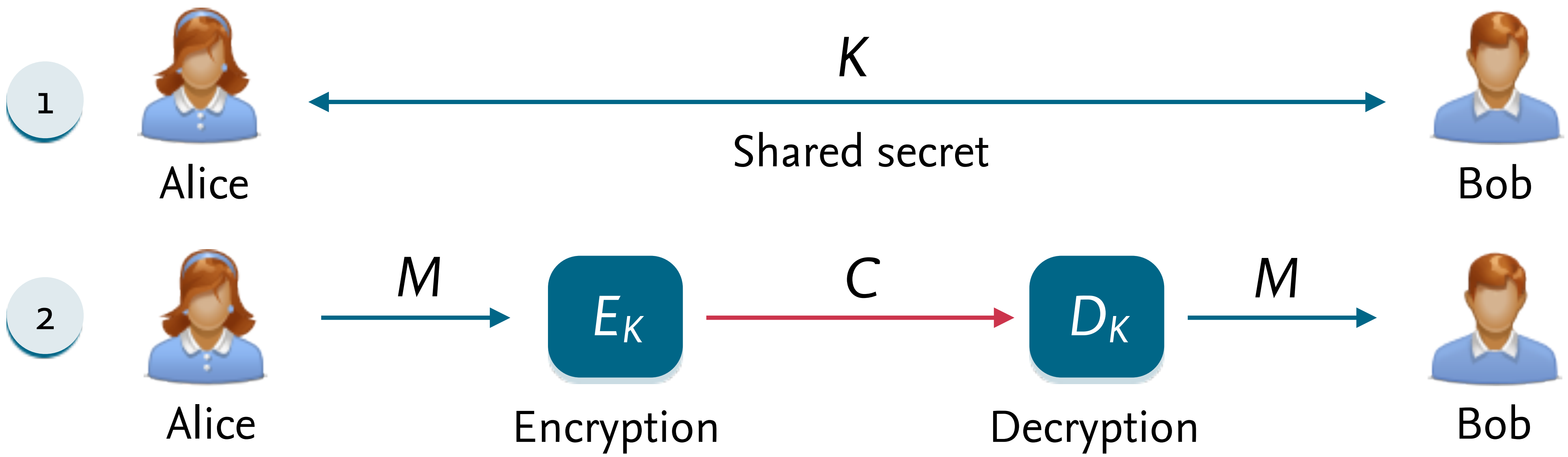
Prof. Dr. Konrad Rieck

Part #1

Overview

- **Topic of the unit**
  - Public-key Cryptography
- **Parts of the unit**
  - Part #1: Asymmetric cryptosystems
  - Part #2: Mathematical Prerequisites
  - Part #3: Rivest-Shamir-Adleman algorithm
  - Part #4: Diffie-Hellman key exchange

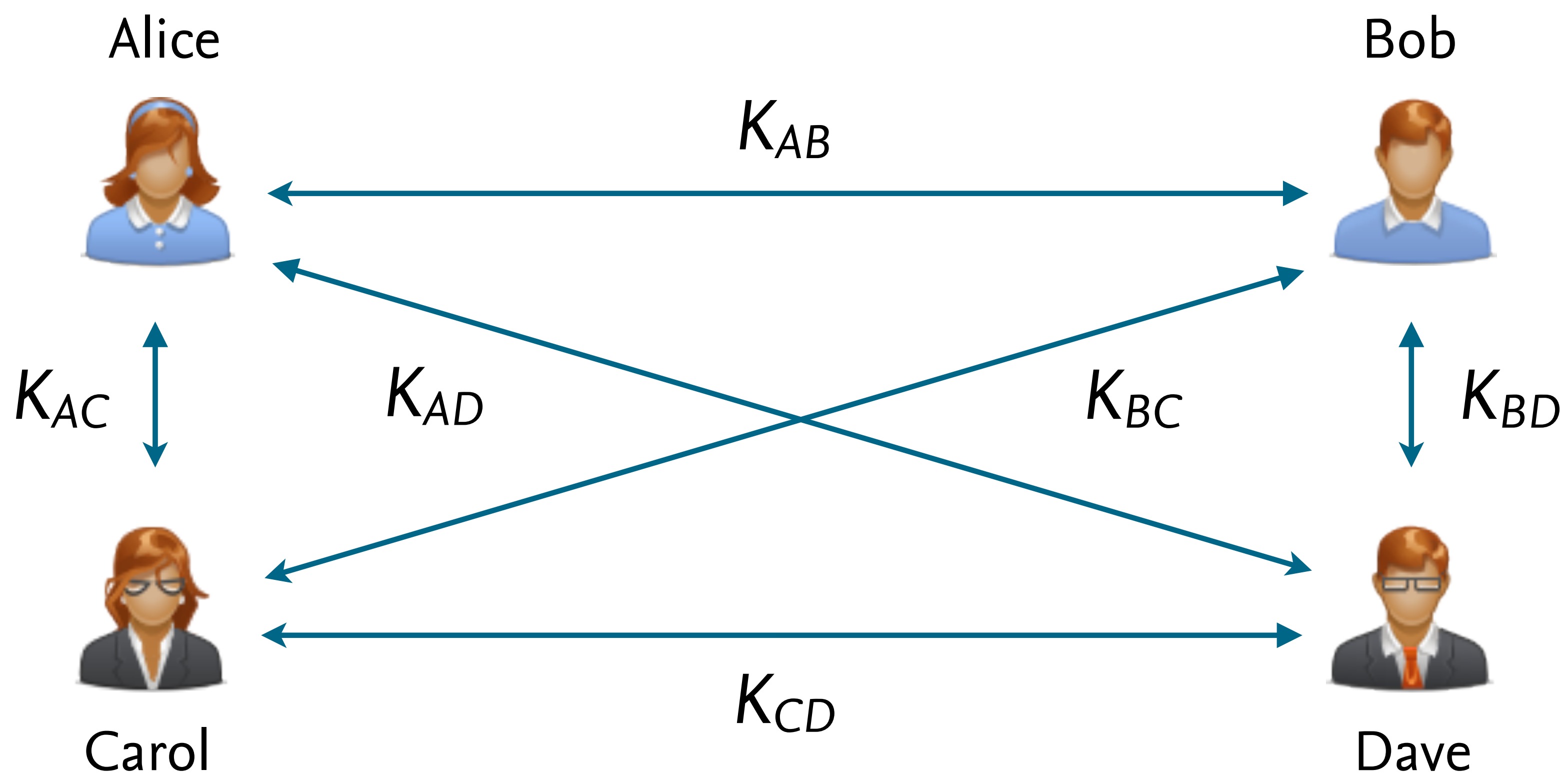
Problem: **Key Exchange**



- **Paradox situation for symmetric cryptosystems**
  - Secure key exchange needed for secure communication
  - Communication between unknown parties not possible

Multi-party Key Exchange

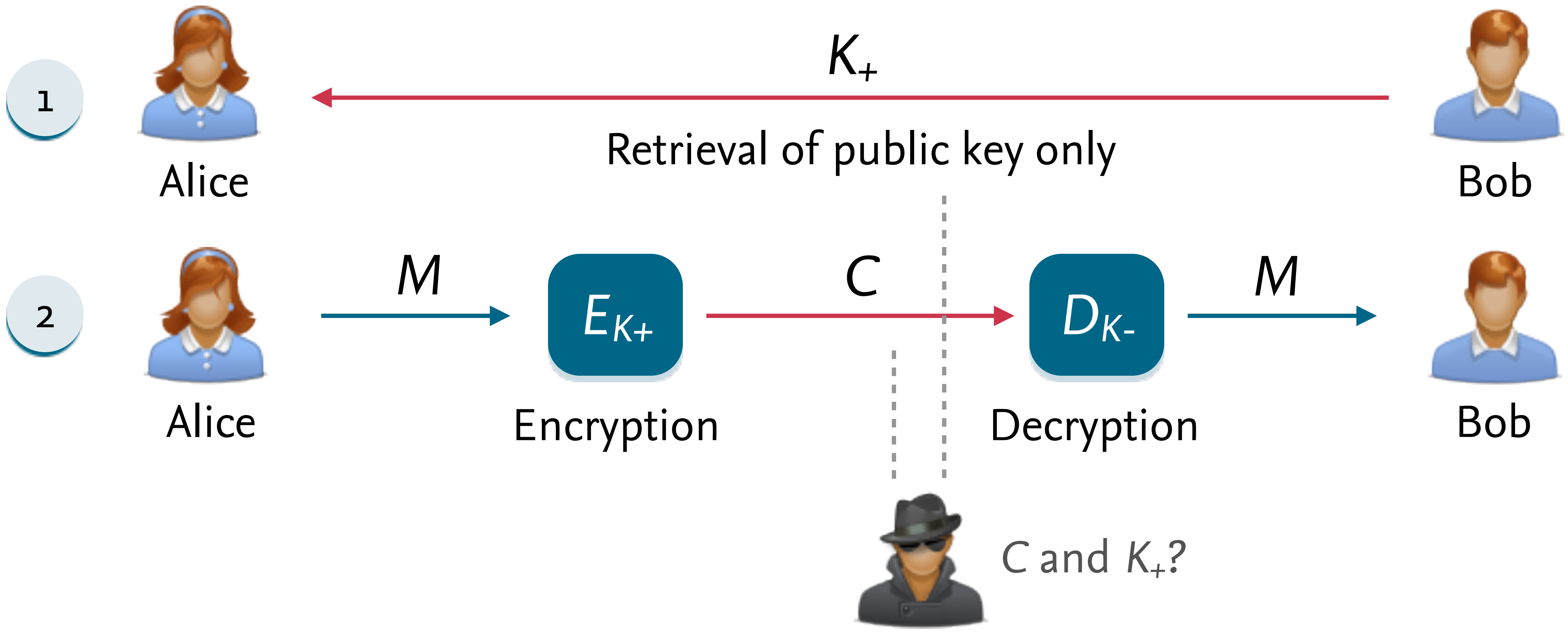
- **Involved multi-party key exchange with symmetric keys**
  - Quadratic growths: n parties → $(n^2 - n) / 2$ keys
  - Problem rooted in symmetry of cryptosystem (shared keys)



Asymmetric Keys

- Solution: **Two types of keys**
  - Public key K_+ = enables encryption but no decryption
  - Private key K_- = used for decryption only
  - Hard to deduce private from public key
- **... similar to a classic mailbox**

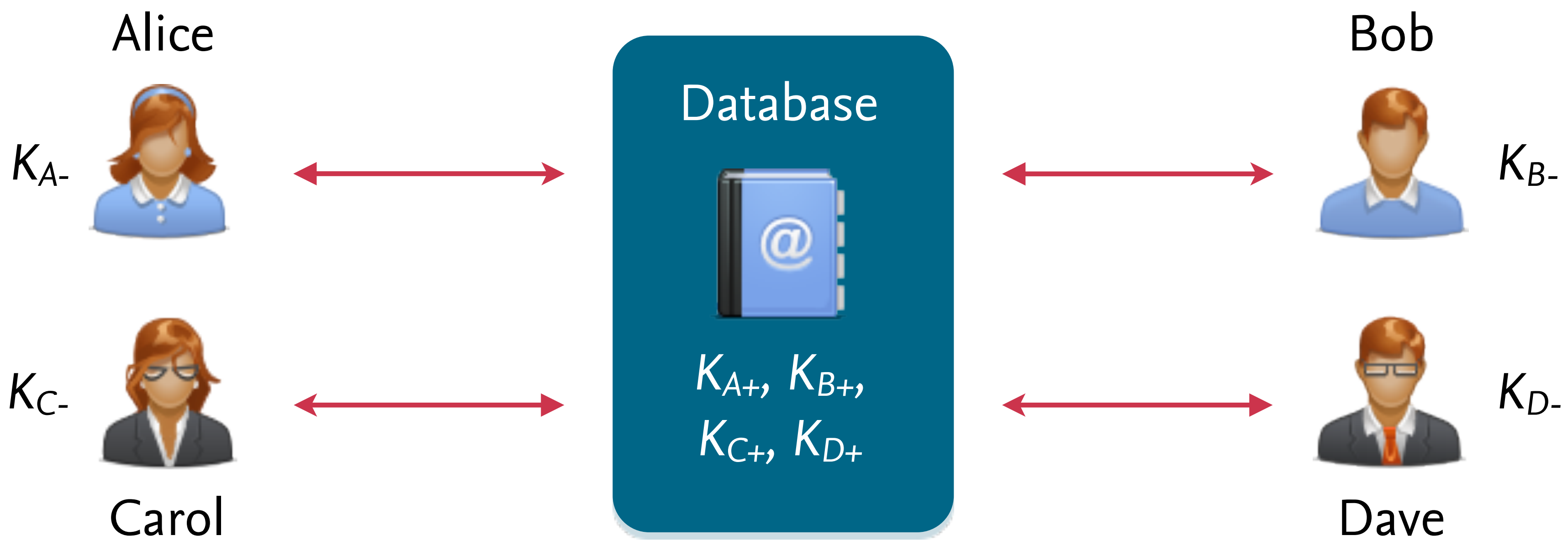
Key Exchange with Public Keys



- **Asymmetric Cipher**
  - K_+ = public key of Bob $K_–$ = private key of Bob
  - No exchange of shared key necessary

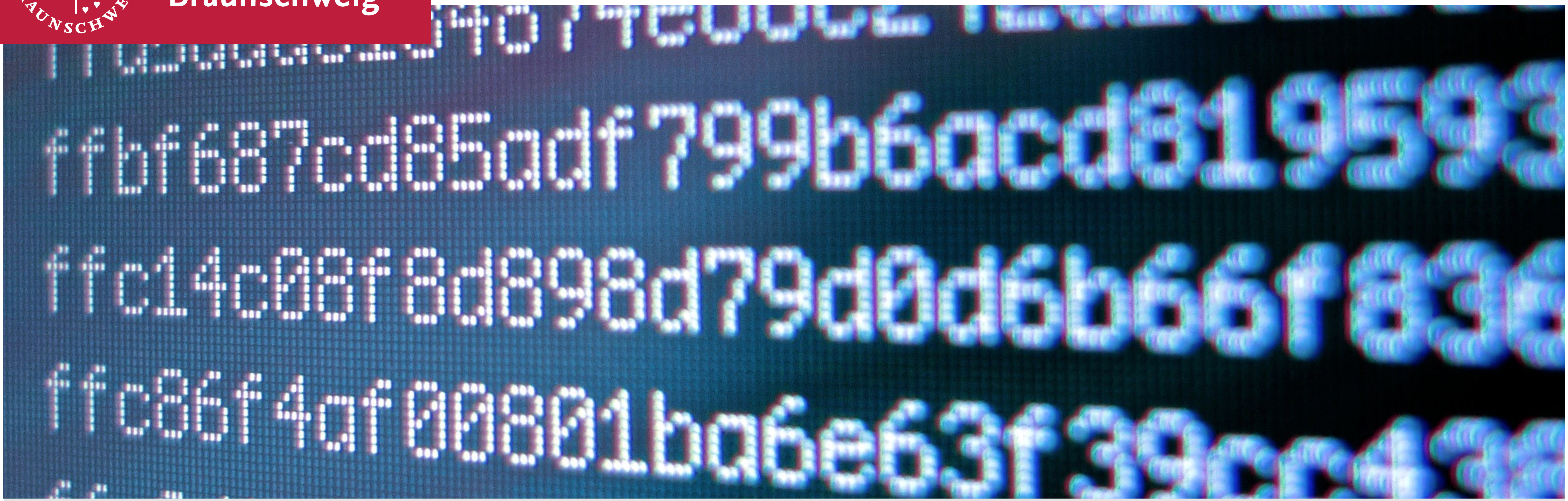
Key Exchange with Public Keys

- **Scalable communication with multiple parties**
  - Linear number of exchanges: *n* parties → *n* public keys
  - Real-world systems with millions of keys (e.g. PGP)



Technische Universität Braunschweig

Institute of System Security



Public-Key Cryptography

Vorlesung “Einführung in die IT-Sicherheit”

Prof. Dr. Konrad Rieck

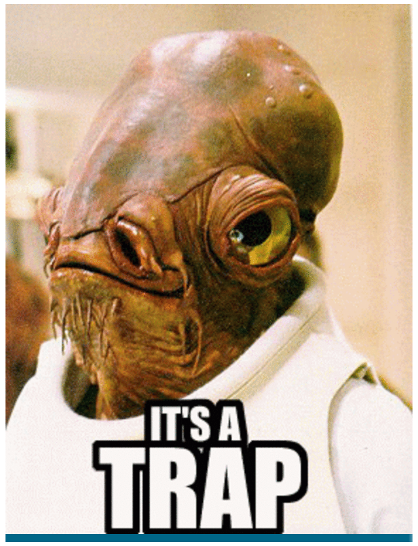
Part #2

Overview

- **Topic of the unit**
  - Public-key Cryptography
- **Parts of the unit**
  - Part #1: Asymmetric cryptosystems
  - Part #2: Mathematical Prerequisites
  - Part #3: Rivest-Shamir-Adleman algorithm
  - Part #4: Diffie-Hellman key exchange

Trapdoor One-Way Functions

- **Trapdoor one-way function** $F(x) = y$
  - Given input *x*: easy to compute output y
  - Given output *y*: hard to compute input *x*
  - Given *y* and a secret: easy to compute *x*



- **Trapdoor basis for asymmetry**
  - Encryption with public key ~ Computing *y*
  - Decryption with private key ~ Computing *x* with secret

- … we need some math for these trapdoors

Modular Arithmetic

- **Modulo operation**
  - Operator determining the remainder of a division
  - For two integers a and n (modulus), we have

$$a \bmod n = b \quad \text{if } a = b + kn$$

- **Modular arithmetic**
  - Arithmetic with integer numbers under a given modulus
  - Examples:

$$\begin{aligned} 11 \equiv 6 \equiv 1 &\pmod 5 \\ 3 + 4 \equiv 2 &\pmod 5 \\ 3 \cdot 5 \equiv 0 &\pmod 5 \\ 2^3 \equiv 3 &\pmod 5 \end{aligned}$$

More Math

- **Greatest common divisor:** $\gcd(a,b) = c$
  - Largest integer c dividing a and b without remainder
  - Computation using factorization or Euclidean algorithm
  - Numbers a and b co-prime if $\gcd(a,b) = 1$
- **Modular multiplicative inverse:** $a \cdot a^{-1} \equiv 1 \pmod{m}$
  - Inverse for modular multiplication
  - Computation using extended Euclidean algorithm
  - Inverse exists only if a and m co-prime

Hard Problems

- **Trapdoor build around a mathematical problem**
  - Problem hard to solve but easy to verify (asymmetry)
  - No polynomial-time algorithm for solution known
  - Examples: Integer factorization and discrete logarithm

> **Integer factorization**
> Given an integer n, find its m prime factors
>
> $$n = p_1 \cdot p_2 \cdots p_m \text{ with } p_i \in \mathbb{P}$$
>
> Example: $n = 4711$ Let's see… $p_1 = 7$, $p_2 = 673$

Hard Problems (cont)

- **Trapdoor build around a mathematical problem**
  - Problem hard to solve but easy to verify (asymmetry)
  - No polynomial-time algorithm for solution known
  - Examples: Integer factorization and discrete logarithm

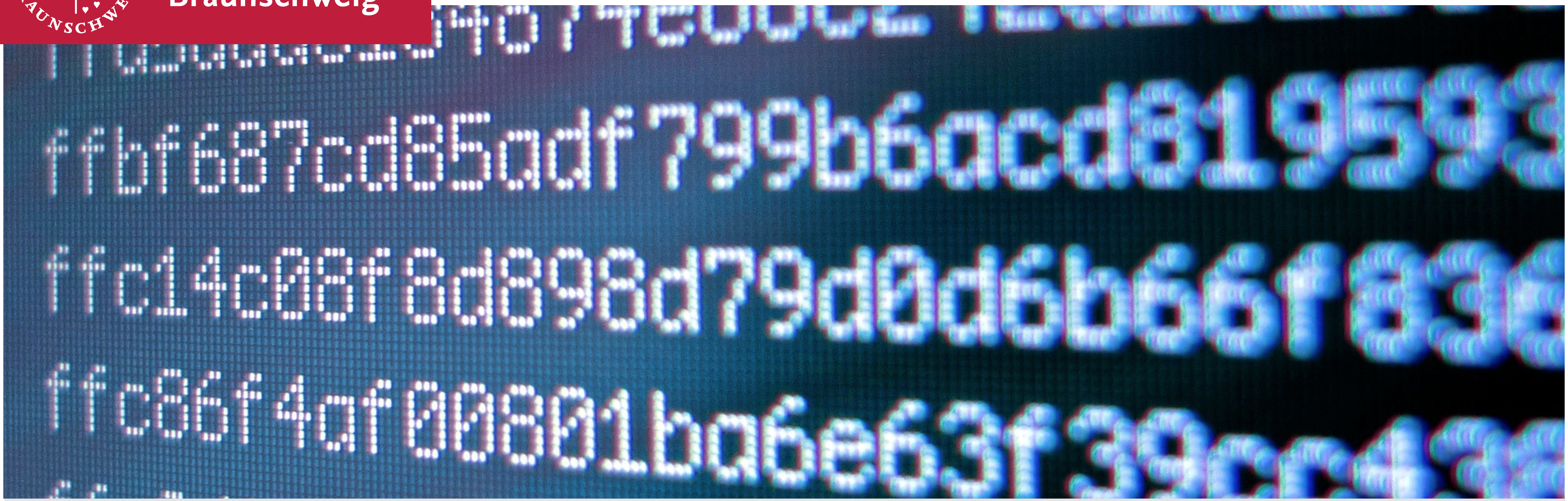
> **Discrete logarithm**
> Given integers g, p, b, find an integer a such that
> $$g^a \equiv b \pmod{p}$$
> Example: $2^a \equiv 1 \pmod{5}$ Let's see... $a = 4$

Asymmetric Algorithms

- **Diffie-Hellman (DH) Key Exchange**
  - Developed by Diffie and Hellman in 1976
  - Based on difficulty of computing discrete logarithms
- **RSA Algorithm** (Encryption & Signing)
  - Developed by Rivest, Shamir and Adleman in 1978
  - Based on difficulty of integer factorization
- **Elgamal Schemes** (Encryption & Signing)
  - Develop by Elgamal in 1985
  - Based on difficulty of computing discrete logarithms

Technische Universität Braunschweig

Institute of System Security



Public-Key Cryptography

Vorlesung “Einführung in die IT-Sicherheit”

Prof. Dr. Konrad Rieck

Part #3

Overview

- **Topic of the unit**
  - Public-key Cryptography
- **Parts of the unit**
  - Part #1: Asymmetric cryptosystems
  - Part #2: Mathematical Prerequisites
  - Part #3: Rivest-Shamir-Adleman algorithm
  - Part #4: Diffie-Hellman key exchange

RSA Algorithm

- **Standard algorithm for public-key cryptography**
  - Developed by Rivest, Shamir and Adleman in 1978
  - Based on difficulty of factorizing large integers

> **Key generation**
> Choose random primes p, q and compute $n = p \cdot q$
> Compute Euler function $\varphi(n) = (p-1)(q-1)$
> Choose random encryption key e with $\gcd(e, \varphi(n)) = 1$
> Compute decryption key $d = e^{-1} \bmod \varphi(n)$

RSA Algorithm (cont)

- **Standard algorithm for public-key cryptography**
  - Developed by Rivest, Shamir and Adleman in 1978
  - Based on difficulty of factorizing large integers

> **Encryption with public key *e*, *n***
> Encrypt message *m* to ciphertext $c = m^e \mod n$

> **Decryption with private key *d***
> Decrypt ciphertext *c* to message $m = c^d \mod n$

RSA Algorithm (cont)

- **Why does RSA work?**

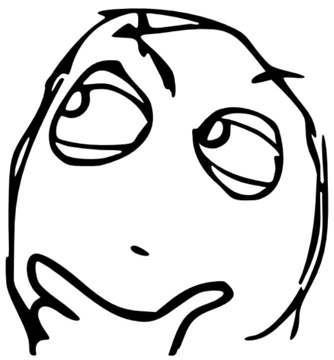
$$c^d \equiv m^{ed} \equiv m^{1+k\varphi(n)} \equiv m \cdot m^{k\varphi(n)} \equiv m \cdot 1 \pmod{n}$$

Substitute c

$d = e^{-1} \bmod \varphi(n)$

That's simple

Euler's theorem

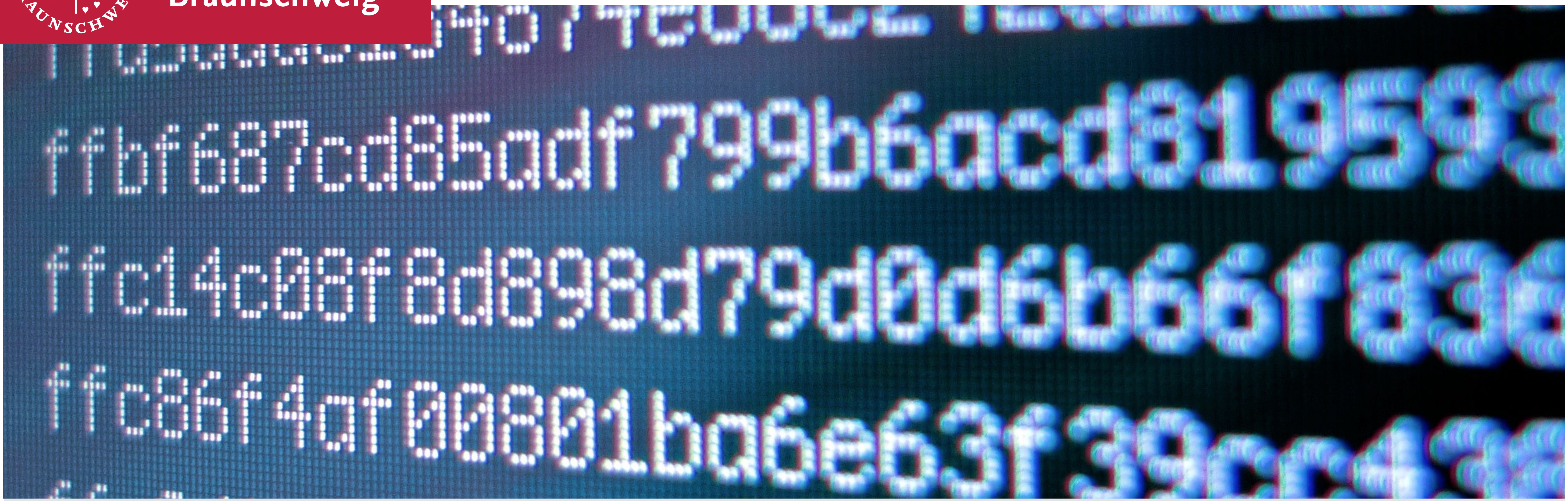


Security of RSA

- **Main attack vectors against RSA**
  - Decrypting ciphertext $c = m^e \bmod n$
    → Difficulty of computing roots in modular arithmetic
  - Deriving private key $d = e^{-1} \bmod \varphi(n)$
    → Difficulty of computing prime factors from n
- **Security depends on size of prime numbers**
  - Factorization of numbers up to 1024 bits feasible
  - Keys with 4096 and more bits considered secure

Technische Universität Braunschweig

Institute of System Security



Public-Key Cryptography

Vorlesung “Einführung in die IT-Sicherheit”

Prof. Dr. Konrad Rieck

Part #4

Overview

- **Topic of the unit**
  - Public-key Cryptography
- **Parts of the unit**
  - Part #1: Asymmetric cryptosystems
  - Part #2: Mathematical Prerequisites
  - Part #3: Rivest-Shamir-Adleman algorithm
  - Part #4: Diffie-Hellman key exchange

Diffie-Hellman Key Exchange

- **Public-key algorithm for secure key exchange**
  - Developed by Diffie and Hellman in 1976
  - Based on difficulty of computing discrete logarithms

> **Initialization**
>
> Alice and Bob agree on prime n and generator g
>
> For all $0 < b < n$ there is an x such that $g^x \bmod n = b$

> **Generation of secrets**
>
> Alice select random number x and sets $X = g^x \bmod n$
>
> Bob selects random number y and sets $Y = g^y \bmod n$

Diffie-Hellman Key Exchange (cont)

- **Public-key algorithm for secure key exchange**
  - Developed by Diffie and Hellman in 1976
  - Based on difficulty of computing discrete logarithms

> **Key exchange**
> Alice sends X to Bob and Bob sends Y to Alice
> Alice computes shared key $k = Y^x \mod n$
> Bob computes shared key $k = X^y \mod n$

Diffie-Hellman Key Exchange (cont)

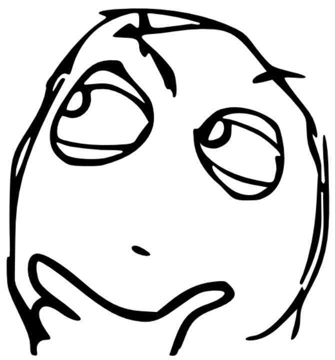
- **Why does the key exchange work?**

$$k \equiv X^y \equiv g^{x^y} \equiv g^{x \cdot y} \equiv g^{y^x} \equiv Y^x \pmod{n}$$

Substitute X

Reverse direction

Logarithm rule



Security of Diffie-Hellman

- **Main attack vectors against key exchange**
  - Determining shared $k = g^{xy} \bmod n$
    → Difficulty of deriving g^{xy} from g^x and g^y mod n
  - Deriving secret number $x = \log_g(X) \bmod n$
    → Difficulty of computing discrete logarithms
- **Security depends on size of X and Y**
  - Difficulty similar to integer factorization
  - Numbers with 4096 and more bits considered secure

Summary

Security and Hard Problems

- **Security of symmetric-key algorithms ⇝ complexity**
  - Diffusion and confusion through involved bit operations
- **Security of asymmetric-key algorithms ⇝ hard problems**
  - Trapdoor property based on hard mathematical problems
  - No polynomial-time solutions *known today*
- **Will these mathematical problems stay hard?**
  - ... advances in quantum computing
  - ... novel polynomial-time algorithms *(or even: $P = NP$?)*

Summary

- **Public-key cryptography**
  - Asymmetric keys → secure key exchange
  - Scalable encryption with multiple parties
- **Security based on trapdoor one-way functions**
  - Integer factorization → RSA algorithm
  - Discrete logarithm → Diffie-Hellman key exchange
  - Security depends on large key sizes (> 3000 bit)