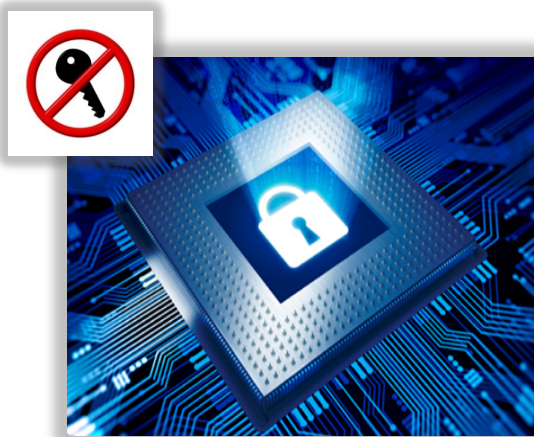
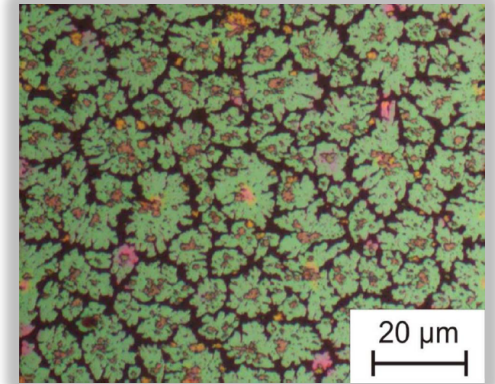




Physical Security: Quantum and Classical



Summer Term
2020



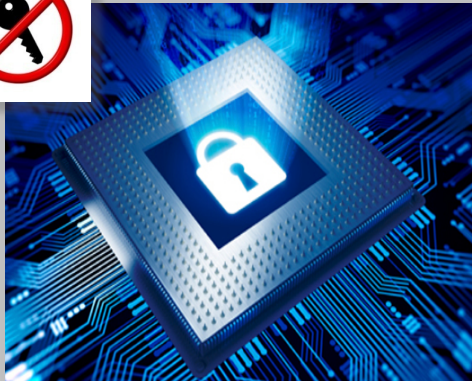
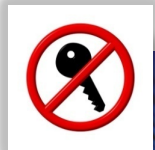
Ulrich Rührmair ^(1,2) and **Harald Weinfurter** ⁽¹⁾

⁽¹⁾ LMU München

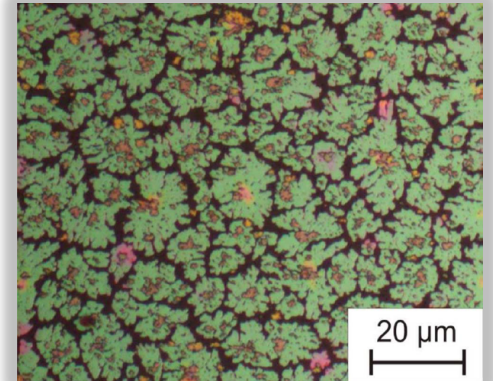
⁽²⁾ University of Connecticut



LECTURE TWO: Introduction to Standard Cryptography (Part II)



April 30,
2020



Ulrich Rührmair

LMU München and
University of Connecticut

Goals of This Lecture

- **Introduction to standard cryptography (part II)**
 - *Asymmetric cryptography,*
aka *public-key cryptography*
 - Cryptographic tasks of
key exchange, public-key encryption,
digital signatures
 - *Three concrete asymmetric cryptographic schemes:*
Diffie-Hellman key exchange,
Rivest-Shamir-Adleman (RSA) public-key encryption scheme,
RSA-based digital signatures
 - Will cover both *conceptual aspects*
and also several *technical details*

Outline

- 1. Computational Complexity of Some Simple Number-Theoretic Problems**
2. Diffie-Hellman Key Exchange
3. RSA Public-Key Encryption Scheme
4. Digital Signatures
5. Conclusions

Computational Complexity

- Some **computational problems** are **more complex** than others...
 - Take more time, consume more energy, need more memory, etc.
 - Practically relevant:
Eats up battery life; heats up your laptop; fan kicks in; etc.
- How can we **formally measure** the „**complexity**“ of a computational problem?

Computational Complexity Cont.

- „Asymptotic“ complexity:
Describe runtime of algorithm **ALG**
(in basic operations/„Turing steps“)
as function $f_{\text{ALG}}(n)$ of the **input length** n
- **Efficient algorithms:**
Those whose runtime is bounded above
by some **polynomial** $p(n)$ („polynomial time“ or „polynomial“ algorithms)
- **Non-efficient algorithms:**
Those whose runtime **cannot** be bounded above
by any polynomial $p(n)$
- **Landau symbols/notation:**
 - $f_{\text{ALG}}(n) = O(g(n))$
if \exists constant C : $f_{\text{ALG}}(n) < C \cdot g(n)$ for all n
 - *Special case of polynomials:* $f_{\text{ALG}}(n) = O(n^k)$
if \exists polynomial of degree k that upper bounds $f_{\text{ALG}}(n)$

Complexity of Some Simple Computational Problems

- **Addition**

- *Schoolbook method:*
Linear no. of steps/
time in input length n
- Complexity: **$O(n)$**

Problem: Addition

Input: Two numbers x , y of length n
(e.g., in binary or decimal notation)

Output: Sum $x + y$
(in same notation)

- **Multiplication**

- *Schoolbook method:*
Quadratic no. of steps/
time in input length n
- Complexity: **$O(n^2)$**
- *Asymptotically best method:*
Harvey-Hoeven-Algorithm ⁽¹⁾, with complexity **$O(n \cdot \log n)$**

Problem: Multiplication

Input: Two numbers x , y of length n

Output: Product $x \cdot y$

- Primality test

- *Schoolbook method:*

- Trial division

- $O(n^2 \cdot 2^{n/2})$



- *Best deterministic method* ⁽¹⁾: $O(n^{7.5})$

- Still not very practical**

- for large numbers x
with large bitlengths $n...$



Problem: Primality test

Input: A natural number x of length n

Output: „Yes“ if x is a prime;
„No“ otherwise

- **Rabin-Miller test** ⁽¹⁾:

- Write $x - 1$
as $x - 1 = 2^t \cdot m$
(where m is odd)

- **Loop k times:**

- Choose random $a \in \mathbb{Z}_x$
- Test if following **condition** is fulfilled:
$$a^m = 1 \pmod{x}$$
or
$$\exists s \in \{0, \dots, t-1\} \text{ such that } a^{2^s m} = -1 \pmod{x}$$
- If condition **not** fulfilled,
then output **„No Prime“** and stop.
Else, repeat **Loop**.

- Output **„Probably Prime“**

Problem: Randomized primality test

Input: A natural number x of length n ,
a natural number k .

Output: „Probably Prime“ or „No Prime“,
with following meaning:

If output is „Probably Prime“, x is a
prime with probability at least $1 - 2^{-k}$
(and no prime with probability
at most 2^{-k}).

If output is „No Prime“,
then x (with certainty!) is no prime.

Complexity of Rabin-
Miller test: $O(k \cdot n^3)$

Problem: Finding uniformly random primes of certain length

Input: Natural numbers n, k

Output: Uniformly random number x of length n , which is prime with probability $1 - 2^{-k}$.

- **Method:**

- Choose uniformly at random a number x of length n
- Test if x is **prime** by **Rabin-Miller test** (with input (x, k))
 - If output is „*Probably Prime*“, accept x as prime (*error probability: 2^{-k}*)
 - If output is „*No Prime*“, repeat with next random x , etc.
- **How often do we need to try?**
Prime number theorem: (no. of primes $\leq n$) $\approx n / \log n$
- Expected runtime complexity: **$O(k \cdot n^3 \cdot \log n)$**

Problem: Factorization

Input: A natural number n

Output: All prime factors p_1, \dots, p_k of n

- **No polynomial time algorithm known.**

Asymptotic runtime complexity
of the best known factoring algorithm (number field sieve):

$$O\left(2^{\left(\sqrt[3]{64/3} + o(1)\right)} \cdot (\ln x)^{1/3} \cdot (\ln \ln x)^{2/3}\right)$$

- Are there **polynomial time algorithms**? We do not know – but most people are convinced that there will not, at least not in our times

Problem: Discrete Logarithm („DiscreteLog“)

Input: Prime p , generator g of \mathbb{Z}_p^* ,
value $g^x \bmod p$

Output: x

- **Asymptotic complexity:** $O\left(2^{\left(\sqrt[3]{64/3} + o(1)\right)} \cdot (\ln x)^{1/3} \cdot (\ln \ln x)^{2/3}\right)$
 - Similar as **Factoring**, albeit with slightly different constants
- **Latest computational record: DiscreteLog for p with 768 bits** ⁽¹⁾
- Are there **polynomial time algorithms**? Again, we do not know – but most people are convinced that there will not, at least not in our times

(1) Fabrice Boudot, Pierrick Gaudry, Aurore Guillevic, Nadia Heninger, Emmanuel Thomé, and Paul Zimmermann:
<https://listserv.nodak.edu/cgi-bin/wa.exe?A2=NMBRTHRY;fd743373.1912>

Some **Provably Not** Efficiently Solvable Problems

- **k-Halting Problem**
(EXPTIME-complete)

Problem: k-Halting Problem

Input: Turing program ⁽¹⁾ P, natural number k

Output: „Yes“ if P halts within k steps,
„No“ otherwise.

- **Halting Problem**
(not even Turing computable,
i.e., cannot be solved
by a fixed computer program
for arbitrary inputs)

Problem: Halting Problem

Input: Turing program ⁽¹⁾ P

Output: „Yes“ if P halts,
„No“ otherwise.

(1) A Turing program can be imagined (**in first order approximation!**) as a finite computer program that is being run on an idealized computer that never breaks or wears off, and which has unbounded memory and energy.

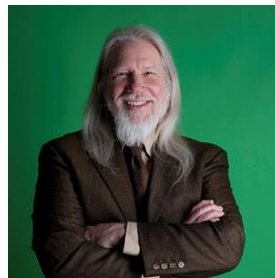
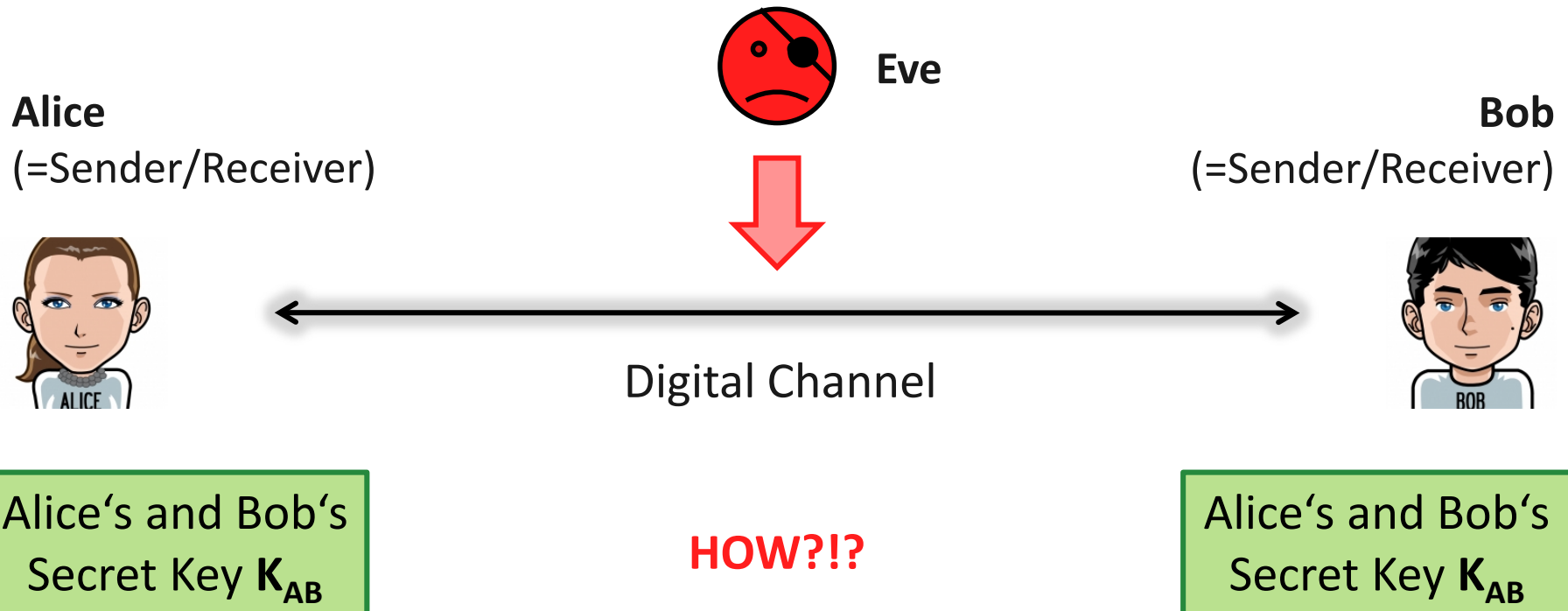
Digest

- Some **computational problems** are more complex than others...
 - **Easy (polynomial time)**: Addition, multiplication, randomized primality testing, finding random primes, **etc.**
 - **Assumed to be hard**:
Factoring numbers, computing discrete logarithms, **etc.**
 - **Provably hard**: k-Halting problem, Halting problem, **etc.**
- Standard measure for **efficient** vs. **non-efficient algorithms**:
 - **Polynomial** vs. **exponential number** of basic computational steps (=Turing steps) in the **length of the input**
- **This lecture**: How to utilize **certain not efficiently solvable** computational problems in cryptography
 - In particular, **DiscreteLog** and **Factoring**

Outline

1. Computational Complexity of Some Simple Number-Theoretic Problems
- 2. Diffie-Hellman Key Exchange**
3. RSA Public-Key Encryption Scheme
4. Digital Signatures
5. Conclusions

Symmetric Encryption: Basic Setting



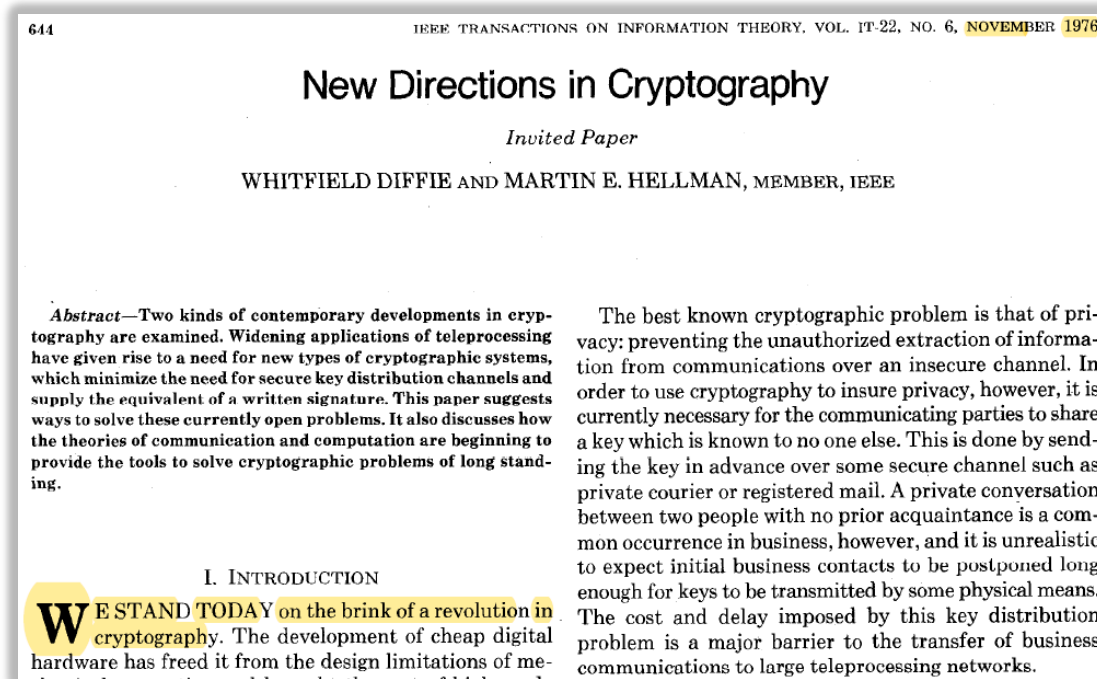
Whitfield Diffie, *1944



Martin E. Hellman, *1945

Whitfield Diffie & Martin E. Hellman, 1976

- **Seminal paper:** „New Directions in Cryptography“



- **Invented Public-Key Cryptography:** Key Exchange Protocols, Public-Key Encryption, Digital Signatures

Diffie-Hellman Key Exchange



Eve (=Attacker/
Eavesdropper)

Alice



$g^a \bmod p$



$g^b \bmod p$



Bob



$$\begin{aligned} K &:= (g^b)^a \bmod p \\ &= g^{ba} \bmod p \\ &= \mathbf{g^{ab} \bmod p} \end{aligned}$$

Public information:

Very large prime p ,
generator g of \mathbb{Z}_p^*

$$\begin{aligned} K &:= (g^a)^b \bmod p \\ &= \mathbf{g^{ab} \bmod p} \end{aligned}$$

- **Practicality:** Finding p and g , and computing g^a , g^b , $(g^b)^a$ and $(g^a)^b \bmod p$, can all be done **efficiently!**



Security of Diffie-Hellman Key Exchange



Eve (=Attacker/
Eavesdropper)

Alice



$g^a \bmod p$



$g^b \bmod p$



Bob



$$\begin{aligned} K &:= (g^b)^a \bmod p \\ &= g^{ba} \bmod p \\ &= \mathbf{g^{ab} \bmod p} \end{aligned}$$

Public information:

Very large prime p ,
generator g of \mathbb{Z}_p

$$\begin{aligned} K &:= (g^a)^b \bmod p \\ &= \mathbf{g^{ab} \bmod p} \end{aligned}$$

- **Security I:** No efficient algorithm for computing $\mathbf{g^{ab} \bmod p}$ from $\mathbf{g^a}$ and $\mathbf{g^b \bmod p}$ is known („Diffie-Hellman or DH-problem“)
- **Security II:** Also no efficient algorithm for computing **discrete logarithms**, i.e., for deriving \mathbf{a} from $\mathbf{g^a \bmod p}$, is known („DiscreteLog problem“)

Security of Diffie-Hellman Key Exchange



Eve (=Attacker/
Eavesdropper)

Alice



$g^a \bmod p$



$g^b \bmod p$



Bob



$$\begin{aligned} K &:= (g^b)^a \bmod p \\ &= g^{ba} \bmod p \\ &= g^{ab} \bmod p \end{aligned}$$

Public information:

Very large prime p ,
generator g of Z_p

$$\begin{aligned} K &:= (g^a)^b \bmod p \\ &= g^{ab} \bmod p \end{aligned}$$

- **Security III:** Unlimited computing power does help Eve!
(\rightarrow DH-KE only has *computational* or *complexity-based* security...)
- **Security IV:** Quantum computers (once/if built) could solve **DiscreteLog**- and **DH**-problem efficiently.

\rightarrow Lectures
3 to 6!

Security of Diffie-Hellman Key Exchange



Eve (=Attacker/
Eavesdropper)

Alice



$g^a \bmod p$



$g^b \bmod p$



Bob



$$\begin{aligned} K &:= (g^b)^a \bmod p \\ &= g^{ba} \bmod p \\ &= \mathbf{g^{ab} \bmod p} \end{aligned}$$

Public information:

Very large prime p ,
generator g of Z_p

$$\begin{aligned} K &:= (g^a)^b \bmod p \\ &= \mathbf{g^{ab} \bmod p} \end{aligned}$$

- **Security V:** DH-problem „reduces“ to **DiscreteLog**.
(I.e., if you can solve **DiscreteLog** efficiently, you can break **DH**).

But not vice versa; not known whether **DiscreteLog** reduces to **DH**.
Both problems hence are **not known** to be generally equivalent. ⁽¹⁾

Diffie-Hellman Key Exchange, **Second Look**

- **Eve in practice**

cannot compute $K = g^{ab} \bmod p$
from knowing g^a and $g^b \bmod p$,
if p is large enough (**≥ 2048 bits**) ⁽¹⁾

*(At least as far as we know,
and if Eve has no large-scale quantum computer).*

- **But:** Is there anything else
that a **smart adversary** can do...?
- *Unfortunately yes:* **Man-in-the-middle attack**

Man-in-the-Middle Attacks



Eve (=Attacker/
Eavesdropper)

Alice



$g^a \bmod p$



$g^b \bmod p$



Bob



$$\begin{aligned} K &:= (g^b)^a \bmod p \\ &= g^{ba} \bmod p \\ &= \mathbf{g^{ab} \bmod p} \end{aligned}$$

Public information:

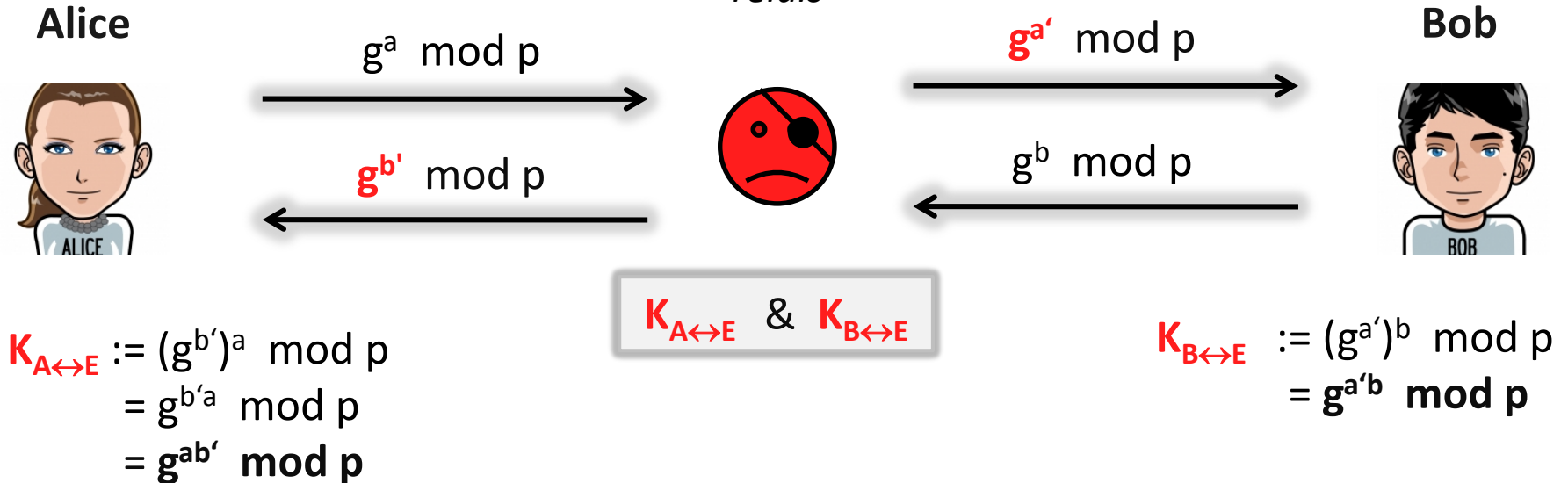
Very large prime p ,
generator g of Z_p

$$\begin{aligned} K &:= (g^a)^b \bmod p \\ &= \mathbf{g^{ab} \bmod p} \end{aligned}$$

- **Idea:** Instead of just **passively** eavesdropping the communication „*from outside*“, **Eve** gets **active**, and **interferes with/alters** communication!

Man-in-the-Middle Attacks Cont.

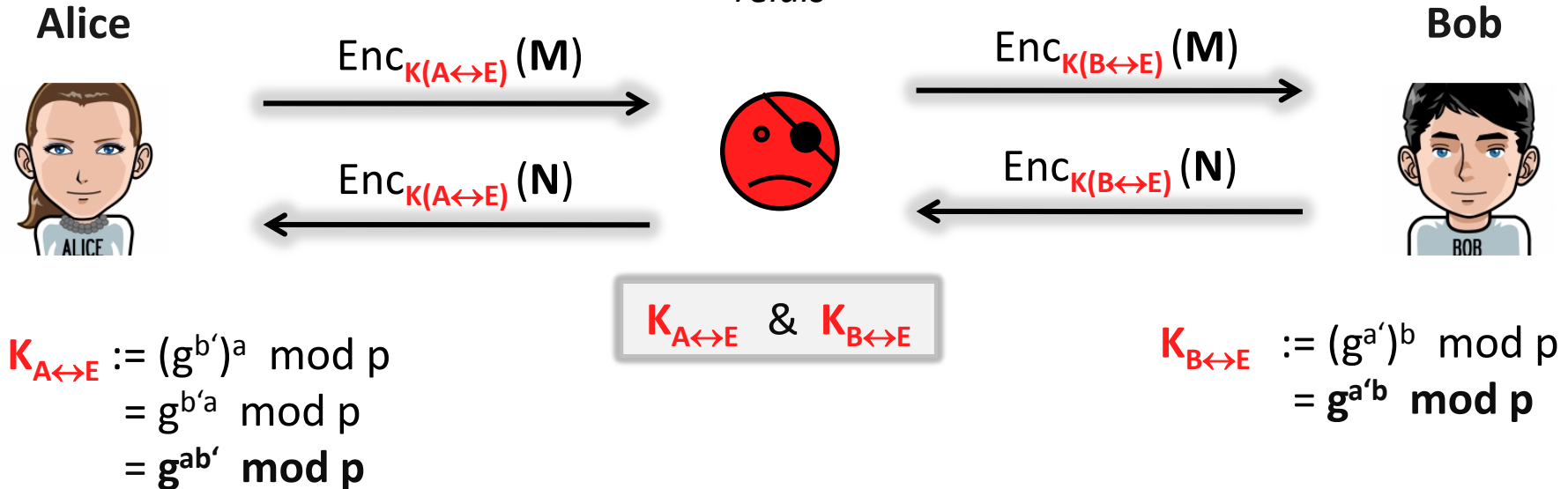
Eve acts as
„communication
relais“



- **Idea:** Instead of just **passively** eavesdropping the communication „*from outside*“, **Eve** gets **active**, and **interferes with/alters** communication!

Man-in-the-Middle Attacks Cont.

*Eve acts as
„communication
relais“*



- **Once Eve** has established key $K_{A \leftrightarrow E}$ with Alice and $K_{B \leftrightarrow E}$ with Bob, she can **forever act as unnoticed „man in the middle“**, reading all messages!

Man-in-the-Middle Attacks, Countermeasure



Eve (=Attacker/
Eavesdropper)

Alice



$g^a \bmod p$

Bob



$g^b \bmod p$

$$\begin{aligned} K &:= (g^b)^a \bmod p \\ &= g^{ba} \bmod p \\ &= g^{ab} \bmod p \end{aligned}$$

Countermeasure:
Authenticate the
communication channel

$$\begin{aligned} K &:= (g^a)^b \bmod p \\ &= g^{ab} \bmod p \end{aligned}$$

- If the communication channel between Alice and Bob is **authenticated**, Eve **cannot** act as man-in-the-middle
- **Authenticated channel** between Alice and Bob is **indeed necessary for any** secure KE protocol: DH, quantum KE, etc.

Man-in-the-Middle Attacks, Countermeasure



Eve (=Attacker/
Eavesdropper)

Alice



$g^a \bmod p$



$g^b \bmod p$



Bob



$$\begin{aligned} K &:= (g^b)^a \bmod p \\ &= g^{ba} \bmod p \\ &= g^{ab} \bmod p \end{aligned}$$

Countermeasure:
Authenticate the
communication channel

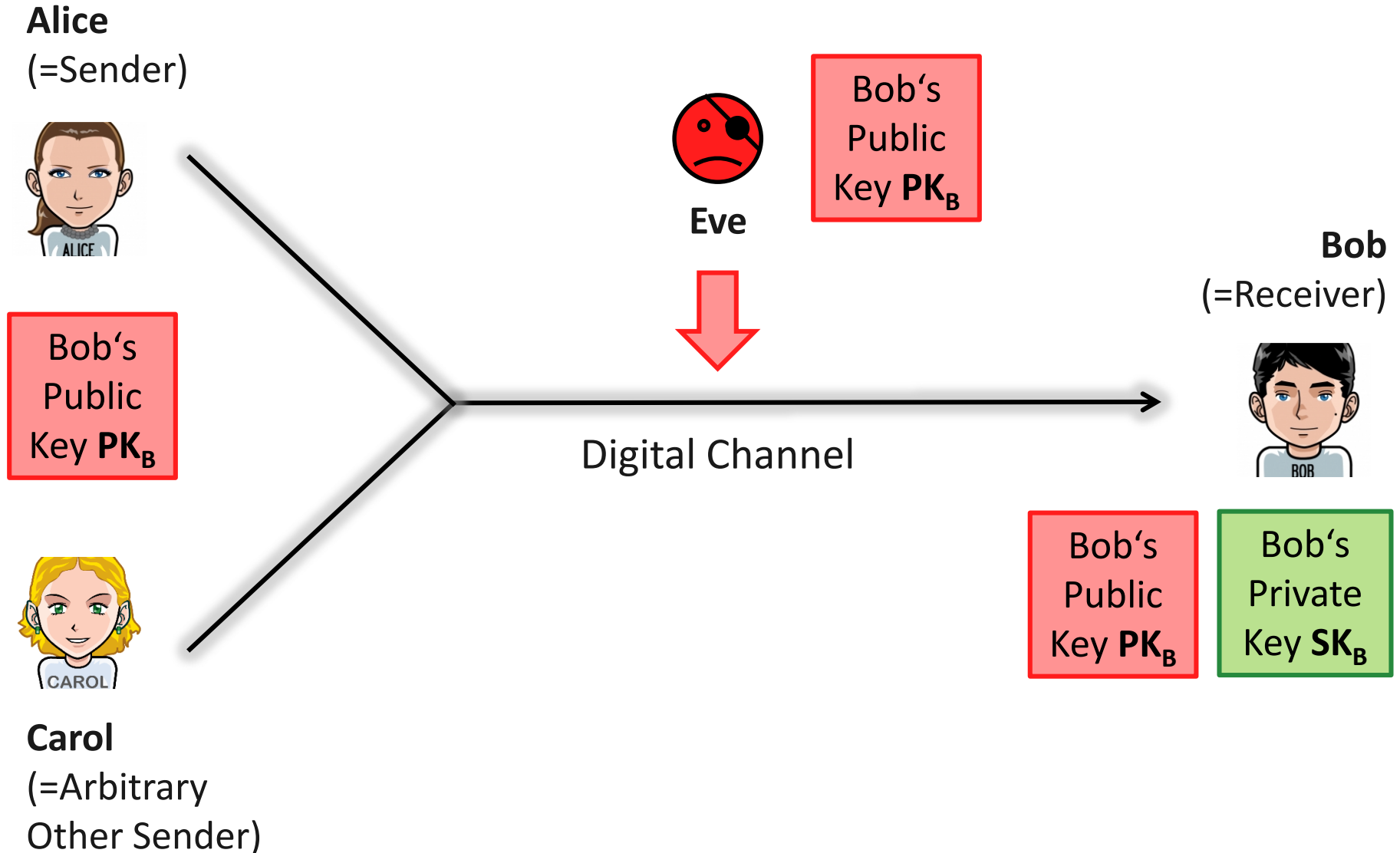
$$\begin{aligned} K &:= (g^a)^b \bmod p \\ &= g^{ab} \bmod p \end{aligned}$$

- **If communication channel is securely authenticated,** and **p is large enough/well chosen, then Diffie-Hellman KE is secure**
 - Possibilities for such authentication: Trusted carrier; trusted postal delivery; small initially shared key (\rightarrow DH as „key amplification/agreement“ scheme); etc.

Outline

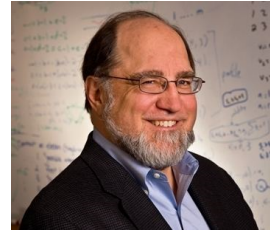
1. Computational Complexity of Some Simple Number-Theoretic Problems
2. Diffie-Hellman Key Exchange
- 3. RSA Public-Key Encryption Scheme**
4. Digital Signatures
5. Conclusions

Asymmetric/Public-Key Encryption: Basic Setting

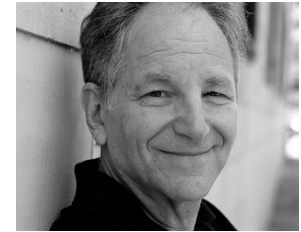


The RSA Scheme (1)

- Published by Ron Rivest, Adi Shamir, Leonard Adleman in 1978 (1)



Ron Rivest, *1947



Leonard Adleman,
*1945



Adi Shamir, *1952

- Splits in three parts:
 - Initialization and *(private key, public key)-pair* generation
 - Encryption method *(using **only** public key)*
 - Decryption method *(using private key)*

Initialization of RSA Scheme

(= *Generation of (Public Key, Private Key)-Pair*)

- Choose **two large primes p and q** of roughly same size independently and u.a.r.
- Compute **$N = p * q$** and **$\varphi(N) = (p-1)(q-1)$**
 - *Note: $\varphi(N)$ is Euler's φ -function, i.e., the order of Z_N^**
- Choose random **encryption exponent e** that is coprime to $\varphi(N)$
 - *Note: One popular choice is $e = 2^{16} + 1 = 65,537$.*
- Compute **decryption exponent d** such that **$e * d = 1 \pmod{\varphi(N)}$**
 - *Note: Given e and $\varphi(N)$, d can be computed efficiently by Euclid's algorithm*

Initialization of RSA Scheme

(= *Generation of (Public Key, Private Key)-Pair*)

- Set **public key/encryption key** to be:

$$\text{Public key} = (e, N)$$

- Set **private key/decryption key** to be

$$\text{Private key} = (d, N)$$

Encryption and Decryption with RSA

- Represent **message M** as number in \mathbb{Z}_N
(*splitting up $M = M_1 \dots M_k$, with $M_i \in \mathbb{Z}_N$, if necessary*)
- **Encryption:** Ciphertext = $M^e \bmod N$
- **Decryption:** Plaintext = $(\text{Ciphertext})^d \bmod N$
= $(M^e)^d \bmod N$
= $M \bmod N$ (*since $e * d = 1 \bmod \varphi(N)$*)
- **Encryption** merely requires **public** key (e, N) .
Decryption necessitates **private** key (d, N)
(*at least so is the hope!*)

Graphical Illustration of RSA

Alice
(=Sender)



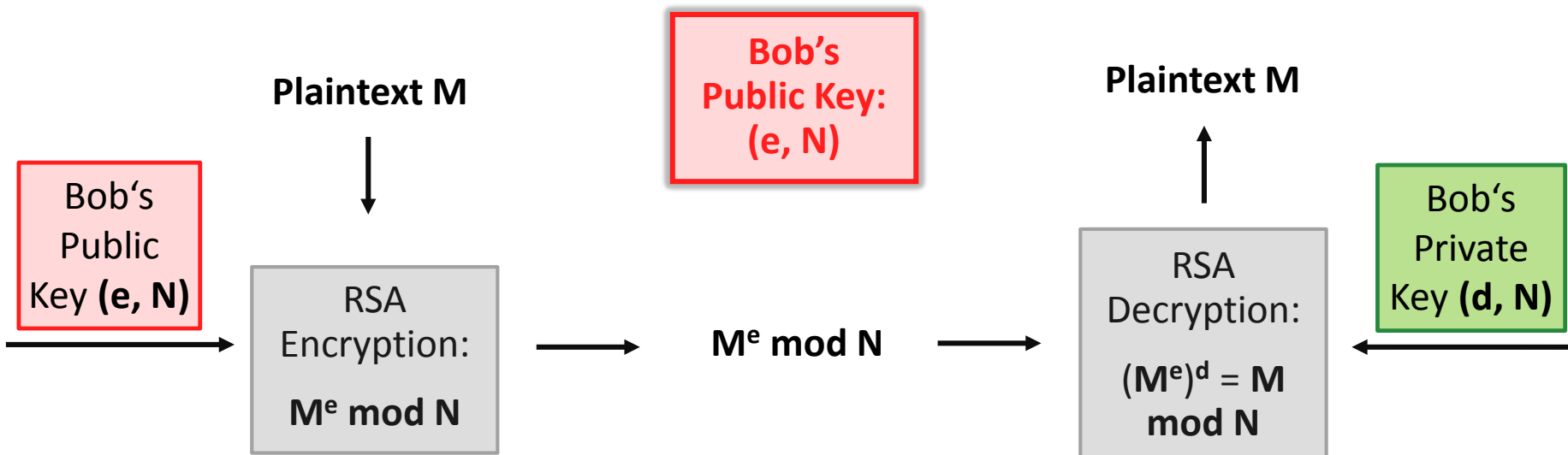
Eve



Bob
(=Receiver)



Digital Channel



Security of RSA: *Computational Security*

- **RSA** „only“ has **computational security**:
Unbounded computational power helps attacker...
 - Given Ciphertext = $M^e \bmod N$,
and public key (e, N) ,
computationally unbounded attacker does:
 - **Factorize** N , obtaining p and q (the two prime factors of N).
Compute $\varphi(N) = (p-1)(q-1)$.
Compute d with $e \cdot d = 1 \bmod \varphi(N)$ (Euclid's algorithm).
Decrypt Ciphertext, using d : $(M^e)^d = M \bmod N$
- For „normal“, not computationally unbounded attackers,
factorization step **assumed too costly** in practice...
(at least without a quantum computer – see Lectures 3 to 6)
- Is **breaking RSA** (=recovering M from $M^e \bmod N$)
equivalent to factoring large numbers? *We do not know...*
 - But currently, no better/other attack than factoring N is known

Security of RSA: *The Case of „RSA-129“*

- **Three inventors of RSA** (Rivest, Shamir, Adleman) in **1977** published in Martin Gardner's *Mathematical Games Column* an RSA-modulus N with **129 decimal digits** („RSA-129“)
- Rivest publicly estimated also in **1977** that **factoring** an RSA-modulus with **125 decimal digits** would take **40 quadrillion years** ($= 4.0 \cdot 10^{16}$ years), on a single computer and latest algorithms available in **1977**
- Already in **1994**, RSA-129 was factored by an **improved factoring algorithm** and **massively parallel computations** distributed all over the internet...
- **Once more: Moore's law**, plus advances in algorithms, plus unforeseen massive computing power in www!
- **Today:** RSA-129 reportedly factored in one day for \$30 on a commercial cloud computing service ⁽¹⁾

RSA Today

- **Key lengths...**

National Institutes of Standards and Technology (NIST):

- **2048-bit** modulus N for RSA-security **until 2030**
- **3072-bit** modulus N for RSA-security **beyond 2030**
- **Implicitly** also suggests that **AES** with **256-bit keys** is “comparable/equivalent” to **RSA** with **15360-bit modulus**.
(But this has to be taken with a lot of care...)

- **Last, but not least:**

Public-key cryptography **more practical** for large networks
in terms of required keys

- *Symmetric crypto:* $N(N-1)/2$ keys (N participants)
- *Asymmetric crypto:* N keys pairs

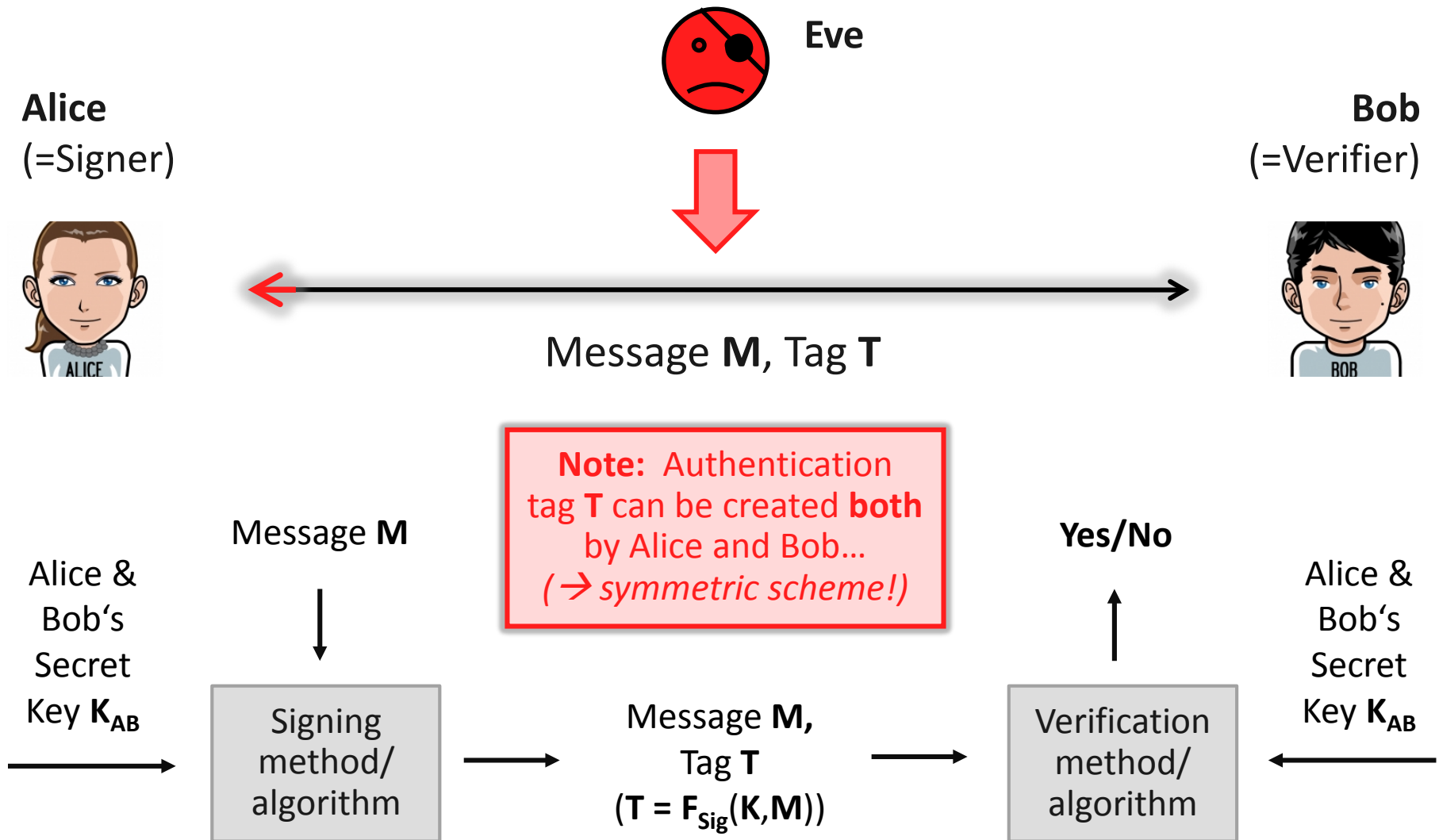
Outline

1. Computational Complexity of Some Simple Number-Theoretic Problems
2. Diffie-Hellman Key Exchange
3. RSA Public-Key Encryption Scheme
- 4. Digital Signatures**
5. Conclusions

Digital Signatures

- Digital signatures...
- ... are nothing else than a **public-key version** (or asymmetric version) of **message authentication schemes!**

Recap Message Authentication: Basic Setting



Digital Signatures: Basic Setting

Alice
(=Signer)



Eve

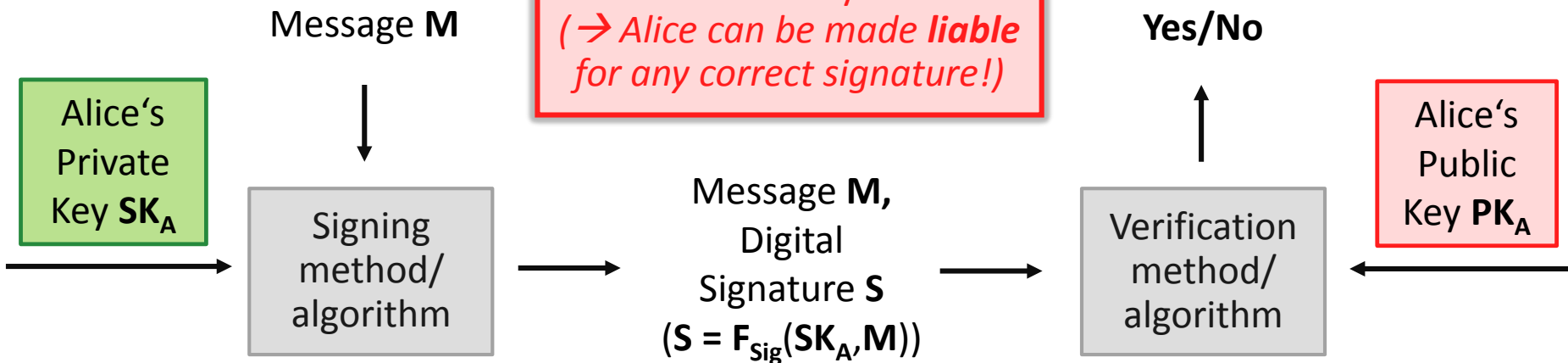
Alice's
Public
Key PK_A

Bob
(=Verifier)



Message M , Digital Signature S

Note: Signature S can only be created by Alice
(\rightarrow Alice can be made *liable* for any correct signature!)



Digital Signatures with RSA

- **RSA** can also be used **for creating digital signatures!**
- Again, represent **message M** as number in \mathbb{Z}_N
(*splitting up $M = M_1 \dots M_k$, with $M_i \in \mathbb{Z}_N$, if necessary*)
- **Alice's private key** (*for signing messages*): $SK_A = (d, N)$
Alice's public key (*for verifying signatures*): $PK_A = (e, N)$
(Note: PK_A is known to Bob, Eve, everyone...)
- **Signing:** Signature $S = M^d \pmod N$
- **Verification:** Given message M and purported signature S' ,
check if $(S')^e = M \pmod N$.

*(Fulfilled for correct S ,
since $S^e = (M^d)^e = M^{de} = M^{ed} = M \pmod N$.)*

Graphical Illustration of Signatures with RSA

Alice
(=Signer)



Eve

Alice's
Public
Key (e,N)

Bob
(=Verifier)



Message M , Signature $M^d \bmod N$

Note: Signature S can only be created by Alice
(\rightarrow liability!)

Message M

Alice's
Private
Key (d,N)

Compute
Signature
 $S = M^d \bmod N$

Message M ,
Signature
 $S = M^d \bmod N$

Yes/No

Check if
 $S^e = M \bmod N$

Alice's
Public
Key (e,N)

General Remarks on RSA-based Signatures

- RSA-based signatures have **computational security**, just like RSA encryption scheme
 - RSA-129 and similar long-term security aspects apply as well
- Similar NIST-recommendations:
 - **2048-bit modulus N** for security until 2030
 - **3072-bit modulus N** for security beyond 2030
 - Also **4096-bit modulus N** recommended by some cryptographers
- In order to **boost efficiency**, and also to **prevent certain attacks**, in practice usually a compactified version or fingerprint of the message **M** is signed (a so-called „**hash value**“ of the message).



Outline

1. Computational Complexity of Some Simple Number-Theoretic Problems
2. Diffie-Hellman Key Exchange
3. RSA Public-Key Encryption Scheme
4. Digital Signatures
- 5. Conclusions**

Summary

- Gave one-lecture overview of **asymmetric cryptography**, aka **public-key cryptography**
- **General cryptographic tasks:** Key exchange, public key encryption, digital signatures
- **Concrete schemes:** Diffie-Hellman key exchange, RSA scheme (both for encryption and digital signatures)
- Presented schemes were mostly based on **elementary number theory** in finite groups
 - Exploited the **assumed practical intractability** of **DiscreteLog** and **Factoring** problems
 - **Alternatives:** Elliptic curves (shorter key lengths at same security level), coding theory, learning with errors, etc.

Long-Term Security in Cryptography

- **Modern cryptography** is undoubtedly an **extremely successful** tool and discipline
- But one potential issue: **Long-term security** (on decades/centuries)
 - **DES** broken by exhaustive key search **ca. twenty years** after introduction (1975/1998)
 - **2006**: Exhaustive key search for DES for \$10,000
 - **RSA-129**: Around **40 quadrillion (=10¹⁵) years** (Ron Rivest) turned into **17 years...** (1977/1994)
 - **2015**: RSA-129 factored in one day for \$30 on commercial cloud computing service...
- **Moore's law** is the systematic reason behind this effect (available computing power for fixed price doubles every 2 years)
- Affects most schemes which merely have **computational security...**

Outlook

- **Next lecture block** will cover (among many other things) how **quantum physics** can help realizing **provable long-term security**
 - **One Idea:** Combine quantum key exchange with one-time pad encryption...
- **Lectures three to six** (*May 7/14/28 & June 4, H. Weinfurter*): **Quantum physics for codemakers and codebreakers**
 - Quantum key exchange and its implementation
 - Quantum computing in cryptographic attacks
 - Quantum random number generation

Further Outlook

- **Lecture seven** (*June 18, guest lecture J.-P. Seifert, TU Berlin*):
Physical attacks on electronic hardware
 - How to extract secret keys from electronic system
 - Invasive attacks, semi-invasive attacks, photonic emission analysis, side channels, backside attacks, etc.
- **Lectures eight to ten** (*June 25 & July 2/9, U. Rührmair*):
Physical Unclonable Functions (PUFs)
 - Weak PUFs and Strong PUFs
 - Implementations and applications of PUFs
 - PUF-Protocols and their security
- **Lectures eleven and twelve** (*July 16/23, U. Rührmair*):
Disorder-Based Security Beyond PUFs
 - Security without secrets
 - Virtual proofs of reality
(i.e., how to remotely prove physical statements over digital communication lines **without** using digital keys at prover)

Thank
you