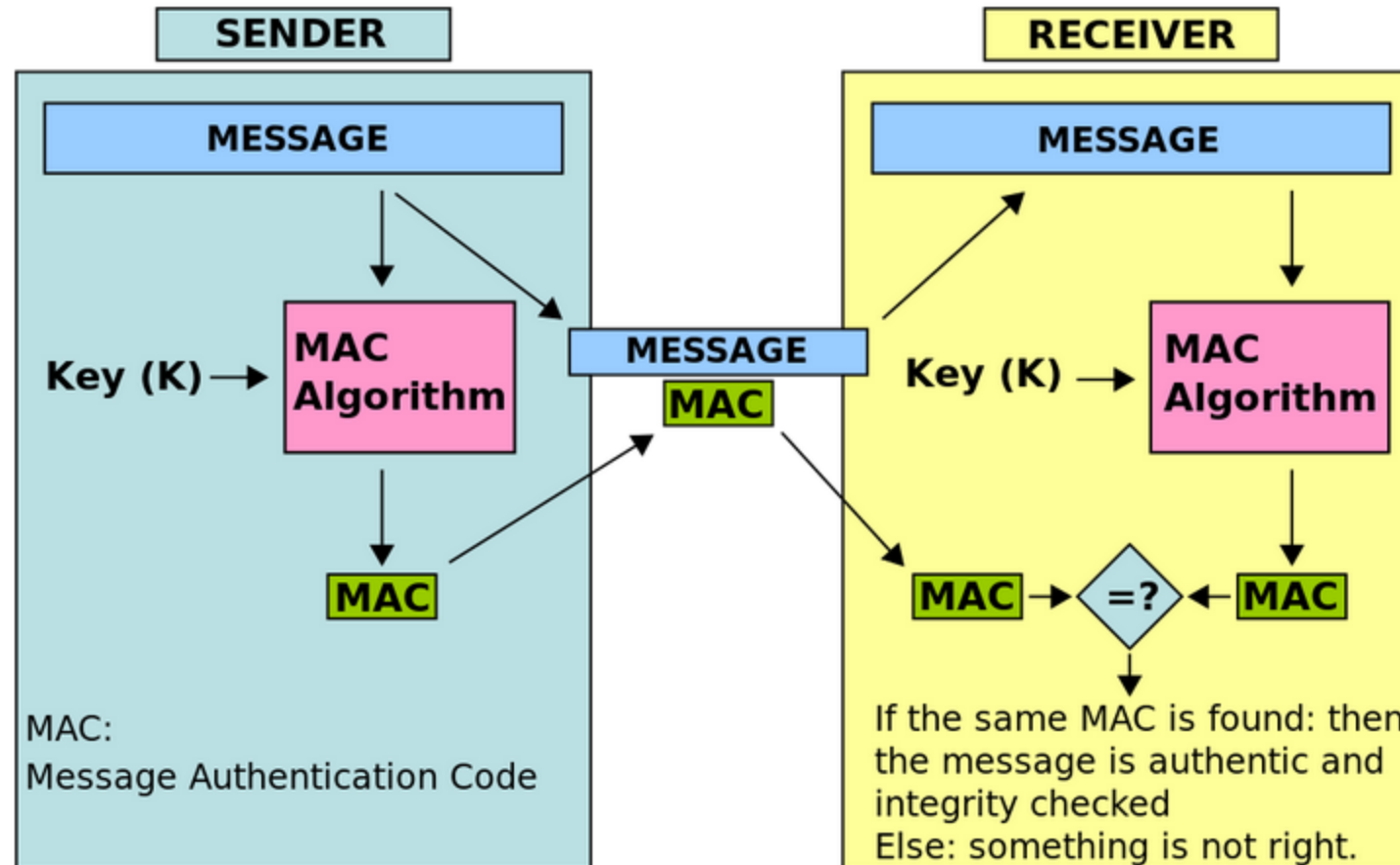




# Kryptographie

Prof. Dr. Björn Grohmann

# MESSAGE AUTHENTICATION CODE



# MESSAGE AUTHENTICATION CODE -- ANGRIFFSVEKTOREN



Hochschule für  
Wirtschaft und Recht Berlin  
Berlin School of Economics and Law

- Total break**      *Alle Systemparameter sind gebrochen*
- Universal forgery (universal unforgeability, UUF)**      *Es kann ein MAC für eine beliebige Nachricht erzeugt werden*
- Selective forgery (selective unforgeability, SUF)**      *Es kann ein MAC für eine Nachricht erzeugt werden, welche vor dem Angriff durch den Angreifer ausgewählt wurde*
- Existential forgery**      *Es kann ein MAC für zumindest eine Nachricht erzeugt werden, welche beliebig sein kann und keinen Sinn ergeben muss*

HOW TO NOT MAC A LONG  
MESSAGE

(WIE MAN AUF KEINEN FALL  
LANGE NACHRICHTEN MIT  
EINEM MAC VERSEHEN  
SOLLTE)

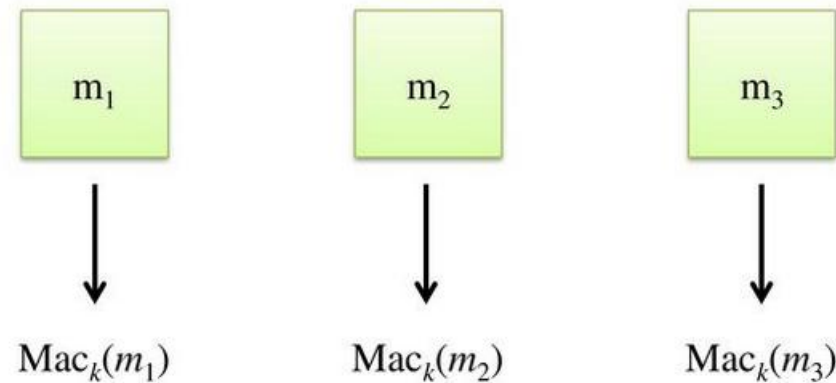


Hochschule für  
Wirtschaft und Recht Berlin  
Berlin School of Economics and Law

# HOW TO **NOT** MAC A LONG MESSAGE



- ❑ **ECB-like construction:** Break a message to fixed-length blocks, and authenticate each individual block.

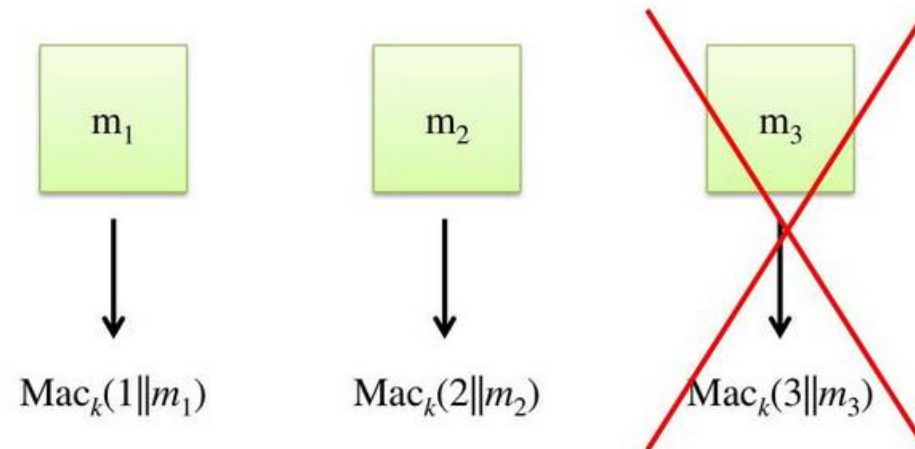


Susceptible to **block re-ordering** attack.

# HOW TO **NOT** MAC A LONG MESSAGE



- ❑ **ECB-like construction with block number:** Like before, but authenticate block numbers as well.

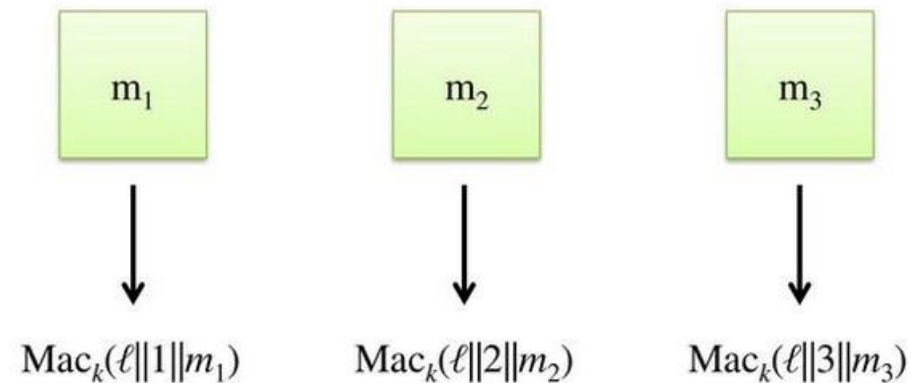


Susceptible to **truncation** attack.

# HOW TO **NOT** MAC A LONG MESSAGE



- ❑ **ECB-like construction with block number and message length:** Like before, but authenticate message length as well.

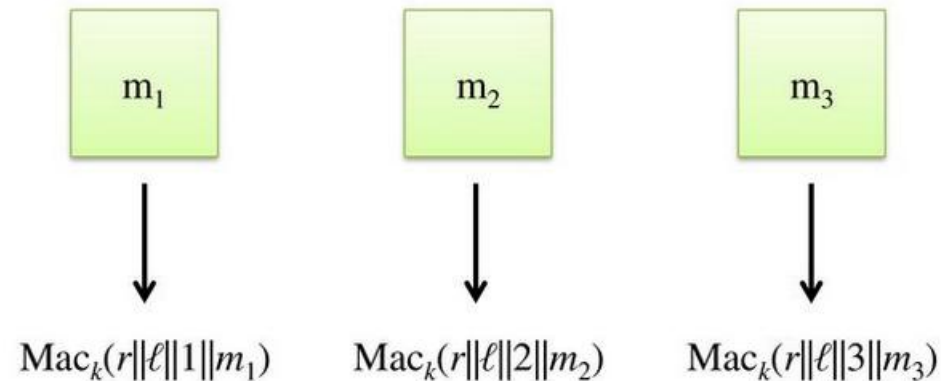


Susceptible to **mix-and-match** attack:  
Forgery by having tags on two *equal-length* messages.

# HOW TO (NOT) MAC A LONG MESSAGE



- Everything like before, but in addition authenticate a **random message identifier**.



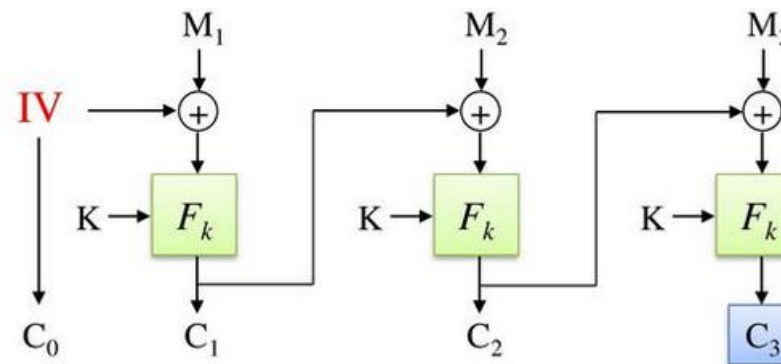
Provably secure construction  
but very inefficient: It **quadruples** tag size!



# HOW TO **NOT** MAC A LONG MESSAGE



- Compute CBC-mode encryption on the message, output the **IV** and the **last block** as the tag.



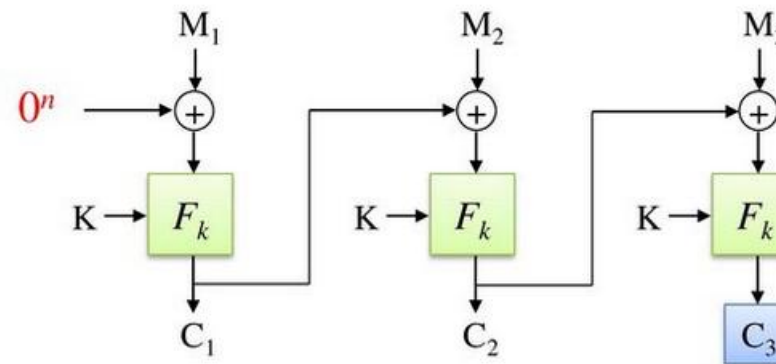
**Forgery:**

$$\begin{aligned} M' &= (M'_1, M_2, M_3) \\ T' &= (IV', C_3) \\ IV' \oplus M'_1 &= IV \oplus M_1 \end{aligned}$$

# HOW TO **NOT** MAC A LONG MESSAGE



- Compute CBC-mode encryption on the message with  $IV = 0^n$ , output the **last block** as the tag.



Susceptible to **length-extension attacks**: Given tag  $C_1$  on single-block message  $M_1$ , output tag  $C_1$  on two-block message  $(M_1, M_1 \oplus C_1)$ .

# BEISPIEL: HASH-MAC

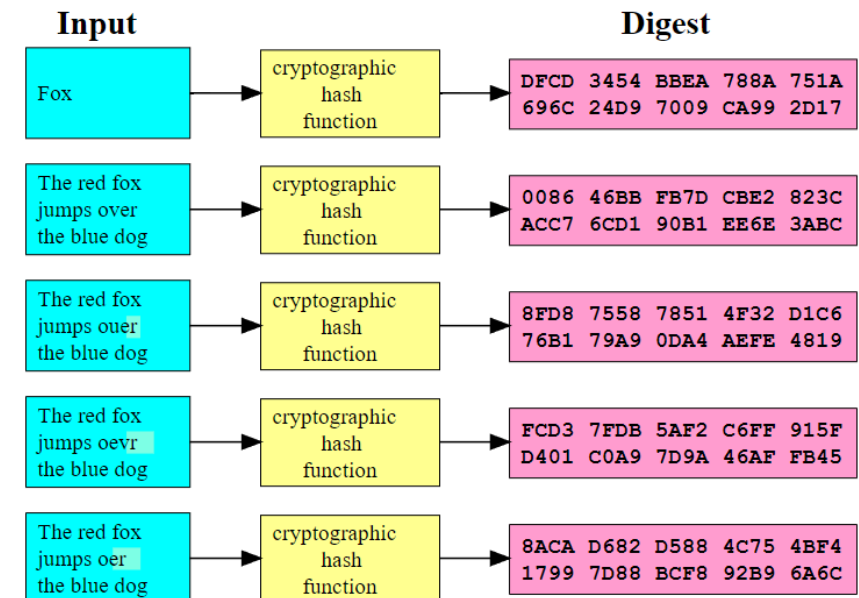


$$\text{HMAC}(K, m) = H \left( (K' \oplus \text{opad}) \parallel H \left( (K' \oplus \text{ipad}) \parallel m \right) \right)$$

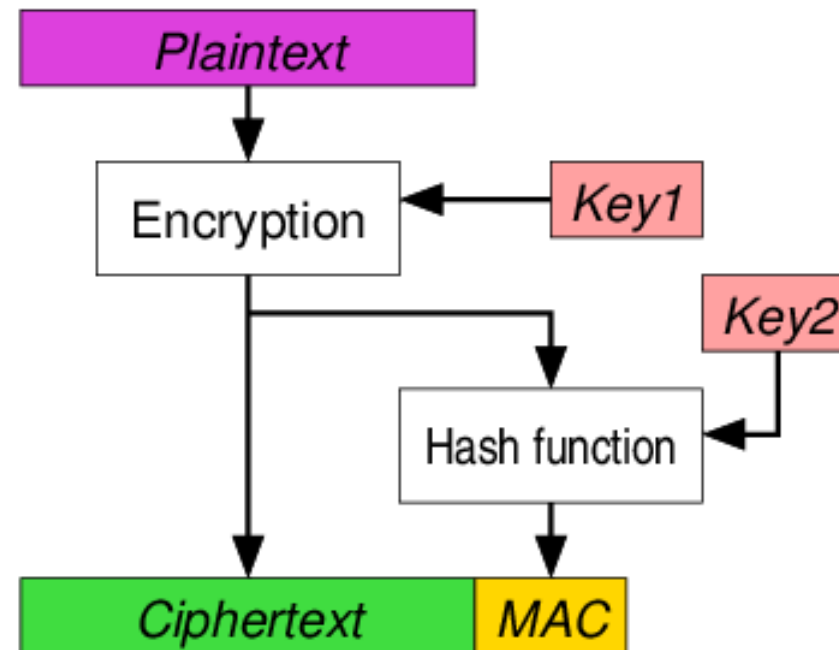
$$K' = \begin{cases} H(K) & K \text{ is larger than block size} \\ K & \text{otherwise} \end{cases}$$

Hierbei ist **H** eine **kryptographische Hash Funktion**:

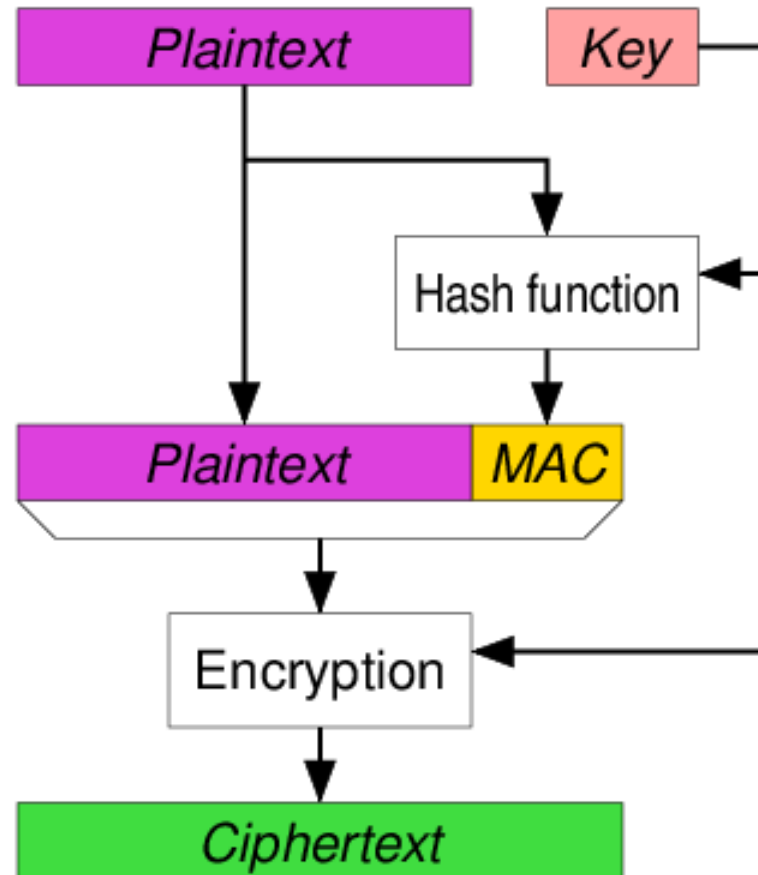
- Effizient zu berechnen
- Schwer zu invertieren
- Schwer eine Kollision zu finden
- Kleine Änderungen bei der Eingabe führen zu großen Veränderungen bei der Ausgabe



# MESSAGE AUTHENTICATION CODE – ENCRYPT-THEN-MAC



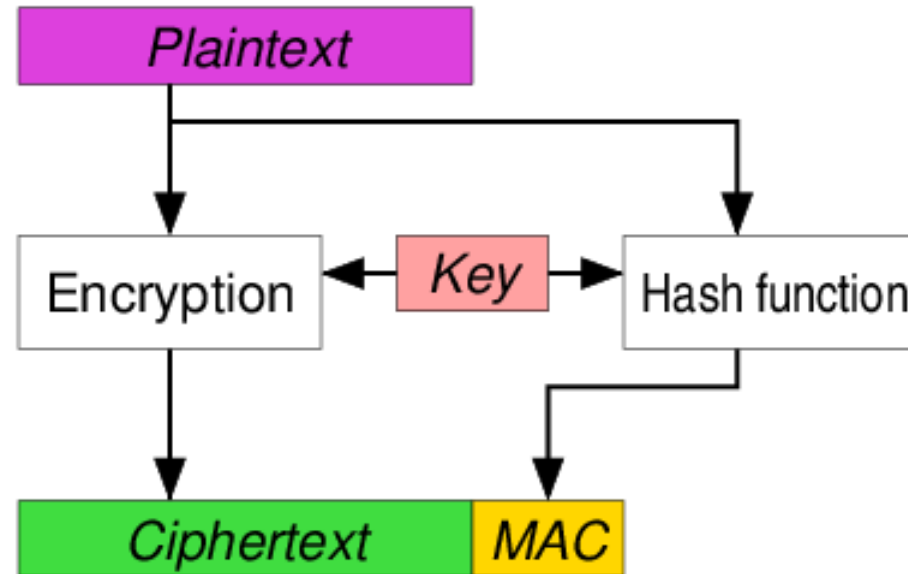
# MESSAGE AUTHENTICATION CODE – MAC-THEN-ENCRYPT



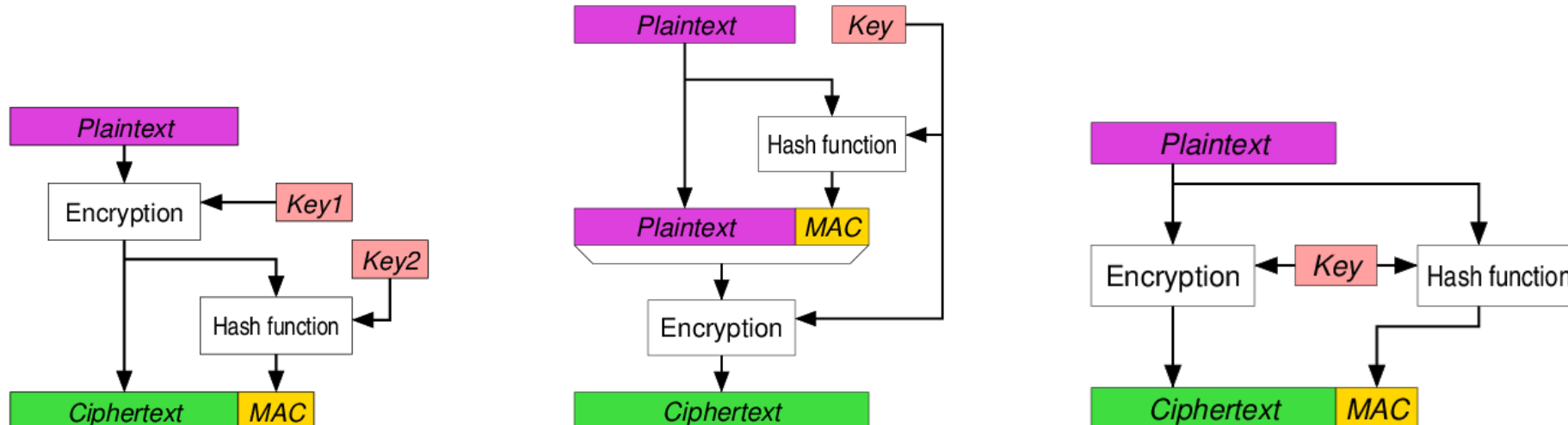
# MESSAGE AUTHENTICATION CODE – ENCRYPT-AND-MAC



Hochschule für  
Wirtschaft und Recht Berlin  
Berlin School of Economics and Law



# MESSAGE AUTHENTICATION CODE



Welche Variante sollen wir jetzt nehmen?

# MESSAGE AUTHENTICATION CODE

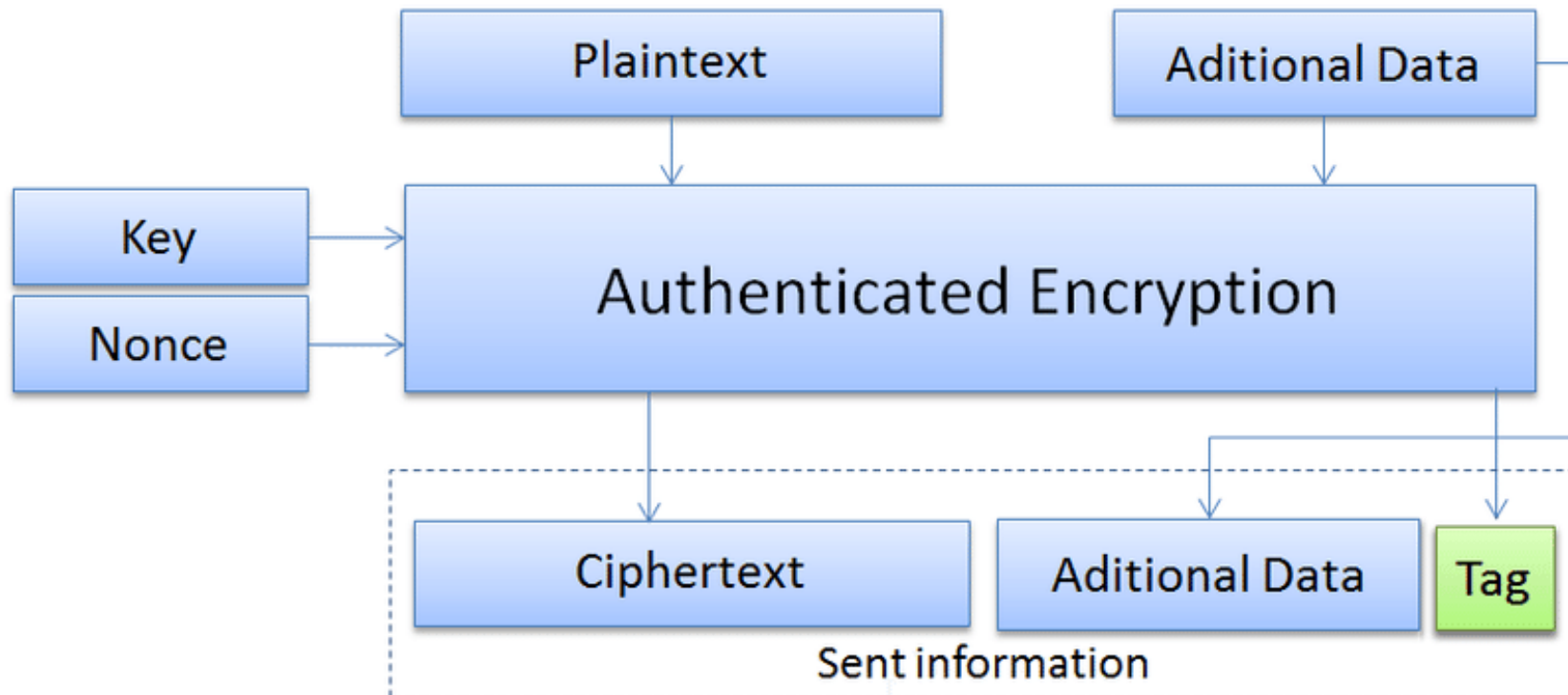


Composition Method	Privacy			Integrity	
	IND-CPA	IND-CCA	NM-CPA	INT-PTXT	INT-CTXT
Encrypt-and-MAC	insecure	insecure	insecure	secure	insecure
MAC-then-Encrypt	secure	insecure	insecure	secure	insecure
Encrypt-then-MAC	secure	secure	secure	secure	secure

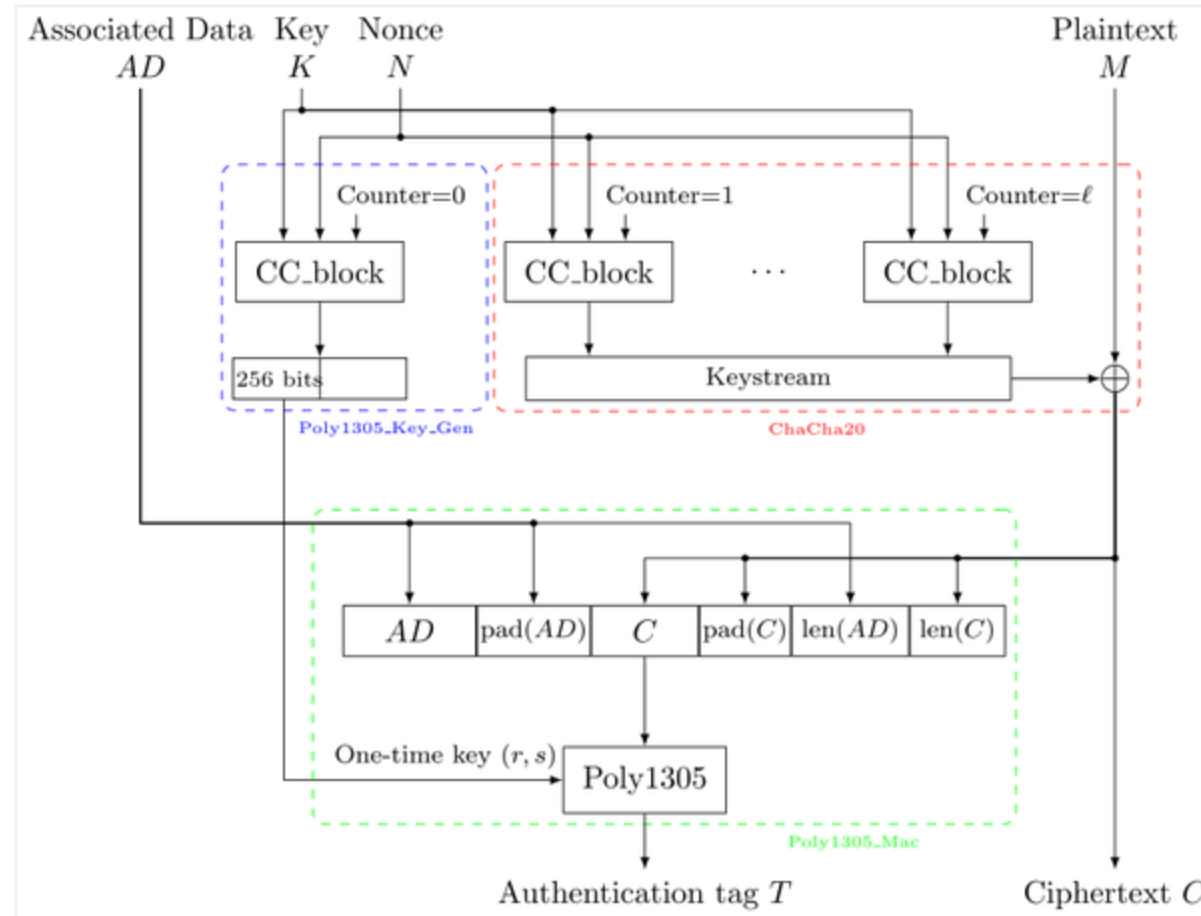
Under the assumption that the MAC scheme is strongly unforgeable



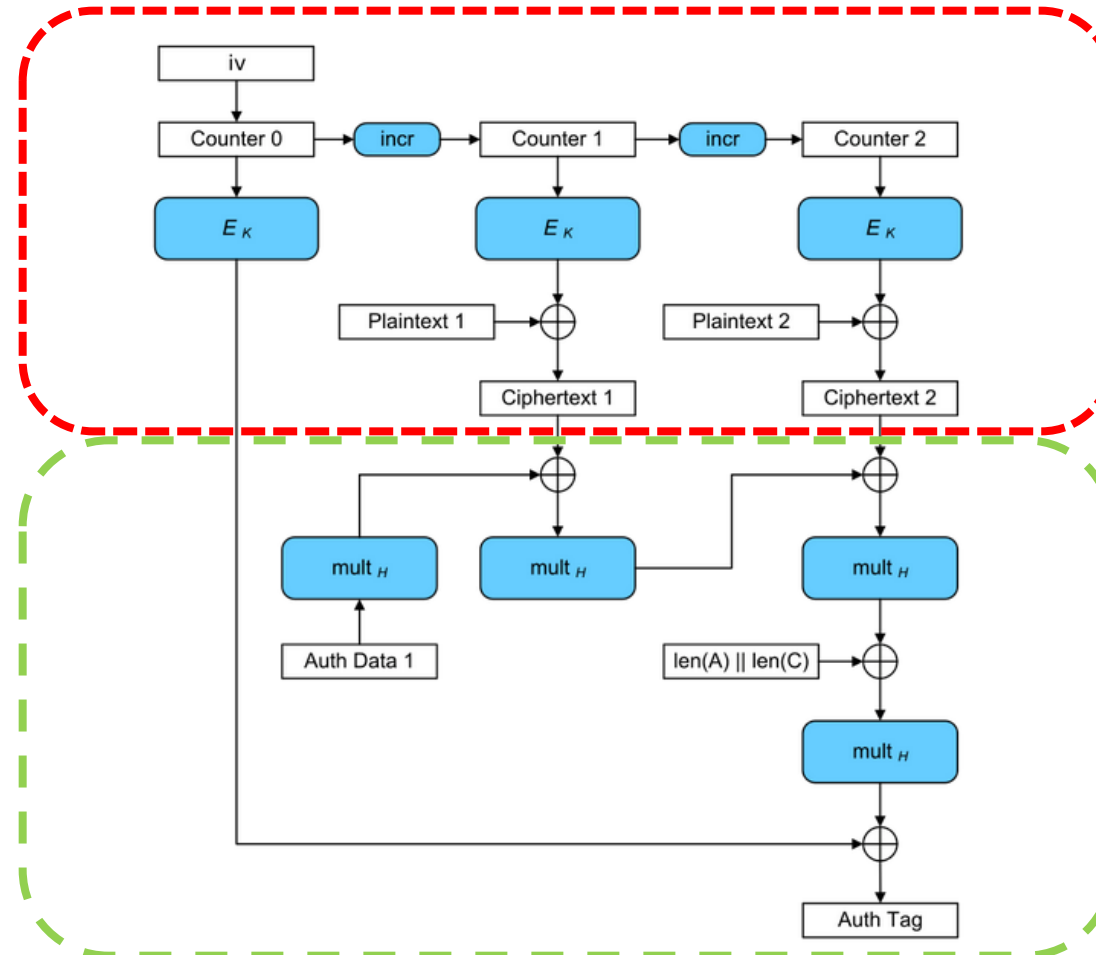
# AUTHENTICATED ENCRYPTION WITH ASSOCIATED DATA (AEAD)



# AEAD BEISPIEL: CHACHA20-POLY1305



# AEAD BEISPIEL: GALOIS COUNTER MODE



ENCRYPTION  
CTR-MODE

MAC

# BEISPIEL: SYMMETRIC SEARCH OVER ENCRYPTED DATA



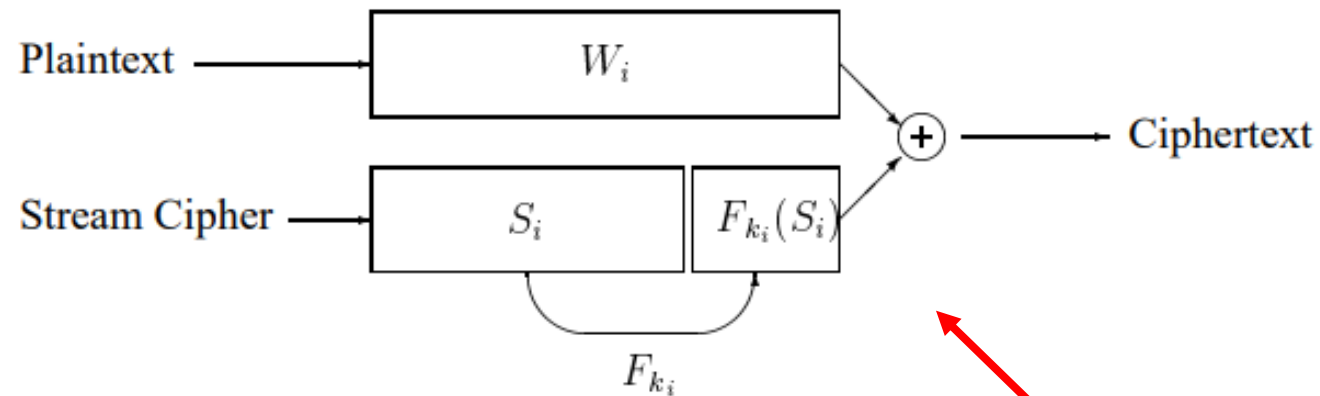
Hochschule für  
Wirtschaft und Recht Berlin  
Berlin School of Economics and Law

*Wie sucht man eigentlich  
auf verschlüsselten Daten?*

# SYMMETRIC SEARCH OVER ENCRYPTED DATA



*erste Idee:*

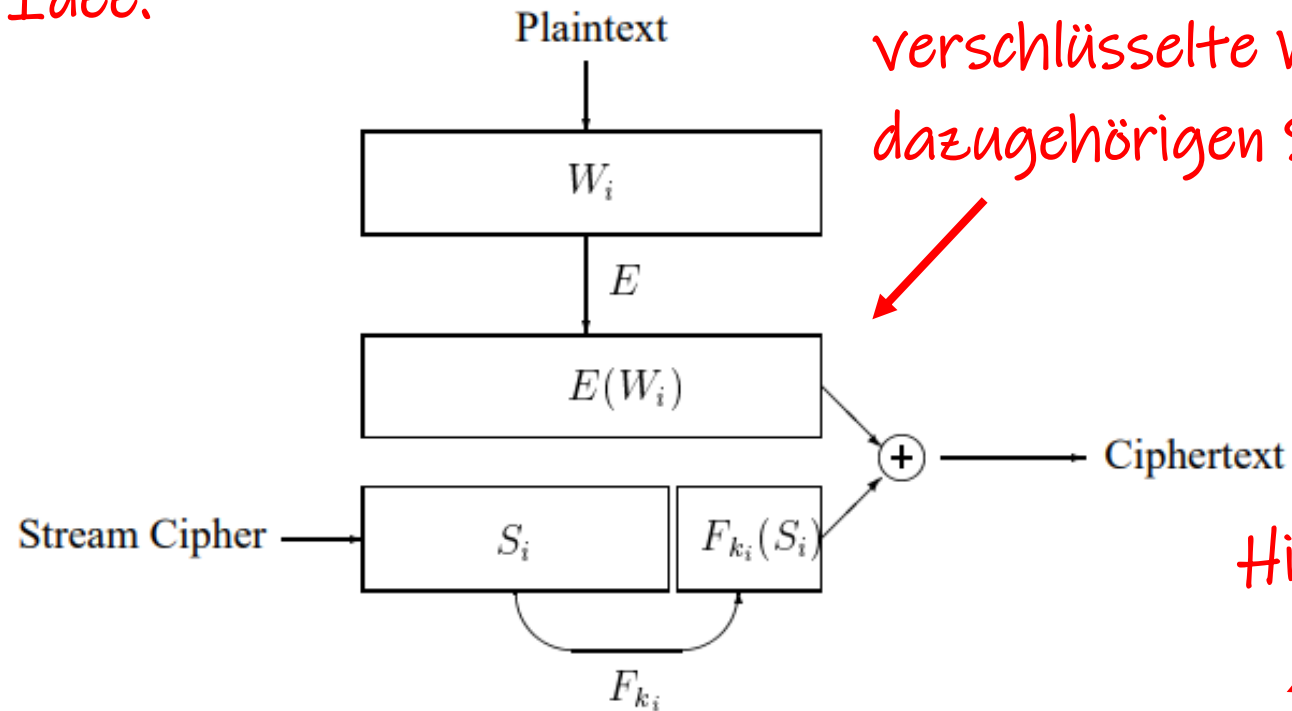


*Für die Suche gibt Alice an Bob das Wort  $W$  und den Schlüssel  $k$*

# SYMMETRIC SEARCH OVER ENCRYPTED DATA



zweite Idee:



Für die Suche gibt Alice an Bob das verschlüsselte Wort  $E(W)$  und den dazugehörigen Schlüssel  $k$

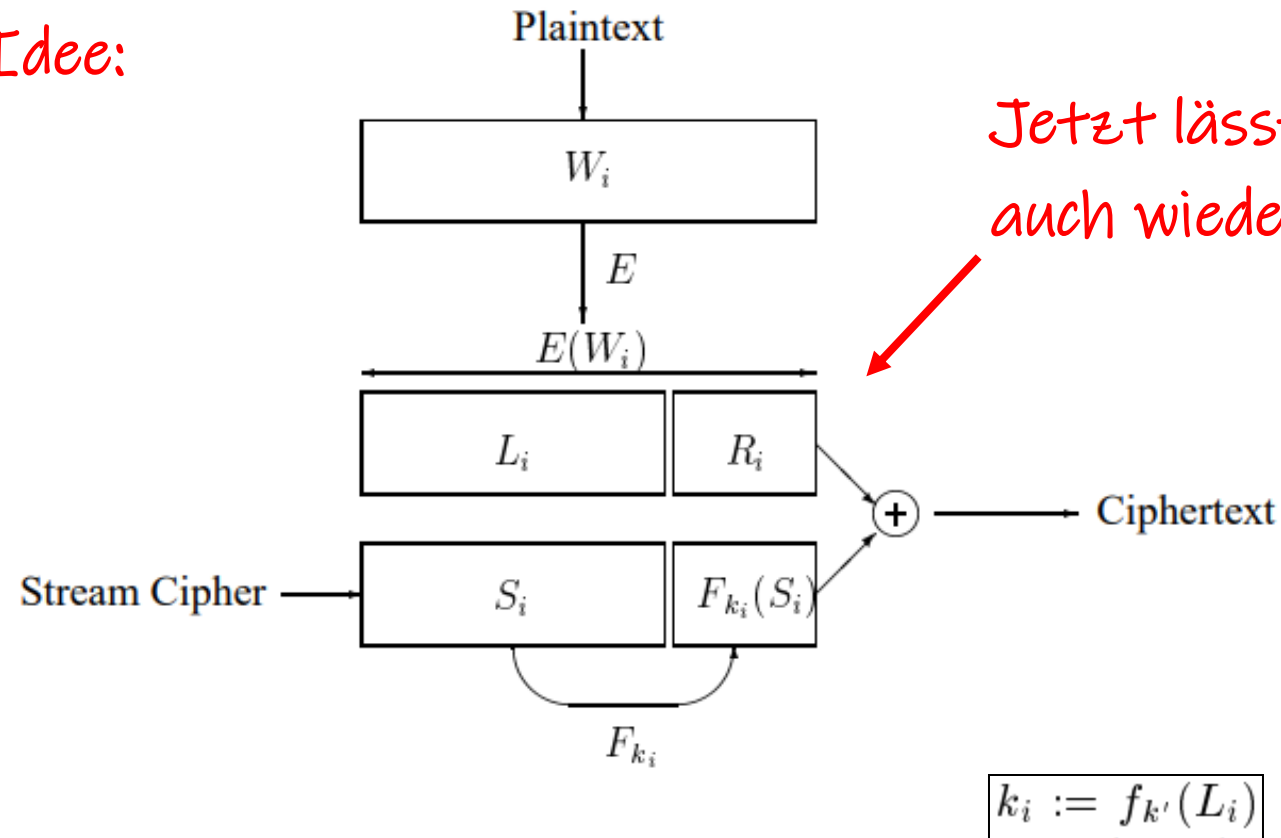
Hier gibt es aber noch einen Haken...

$$k_i := f_{k'}(E_{k''}(W_i))$$

# SYMMETRIC SEARCH OVER ENCRYPTED DATA



*dritte Idee:*



# PUBLIC KEY CRYPTOGRAPHY



Hochschule für  
Wirtschaft und Recht Berlin  
Berlin School of Economics and Law

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



# HOW TO EXCHANGE THIS KEY?

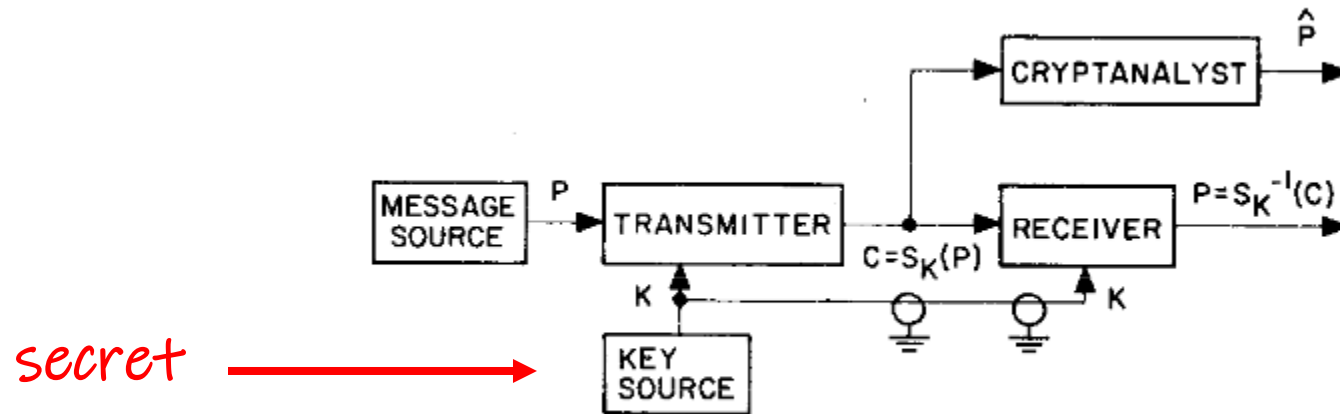
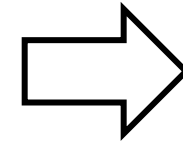


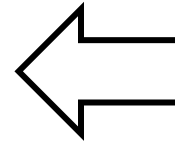
Fig. 1. Flow of information in conventional cryptographic system.

# IDEE: MERKLE PUZZLE

Id:234234234, key: 32748264287482647284723648	shdkfjhskdfhksjhdhfkjshkdfjhskjfhdksh
Id:928372482, key: 87347268268728736482748274	ksjhdfiwuherkkfshdkfjhiuwhefhkwjhfkd
Id:182374674, key: 98293843476287461928383766	wiwebksbfjhgsjhfdsbfsjhffkhskjhfhke
:	:
Id:776268376, key: 23476347268584298374823742	risnbvhzgwuzegfkjsnvjsvgskjfhnsfnfj



Id:4658745683



# PUBLIC KEY CRYPTOGRAPHY

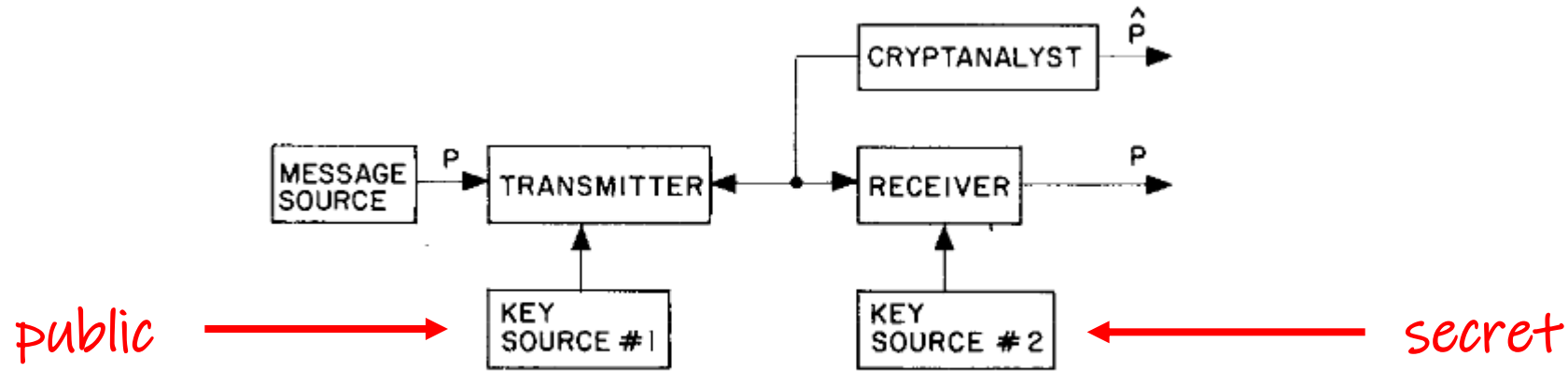
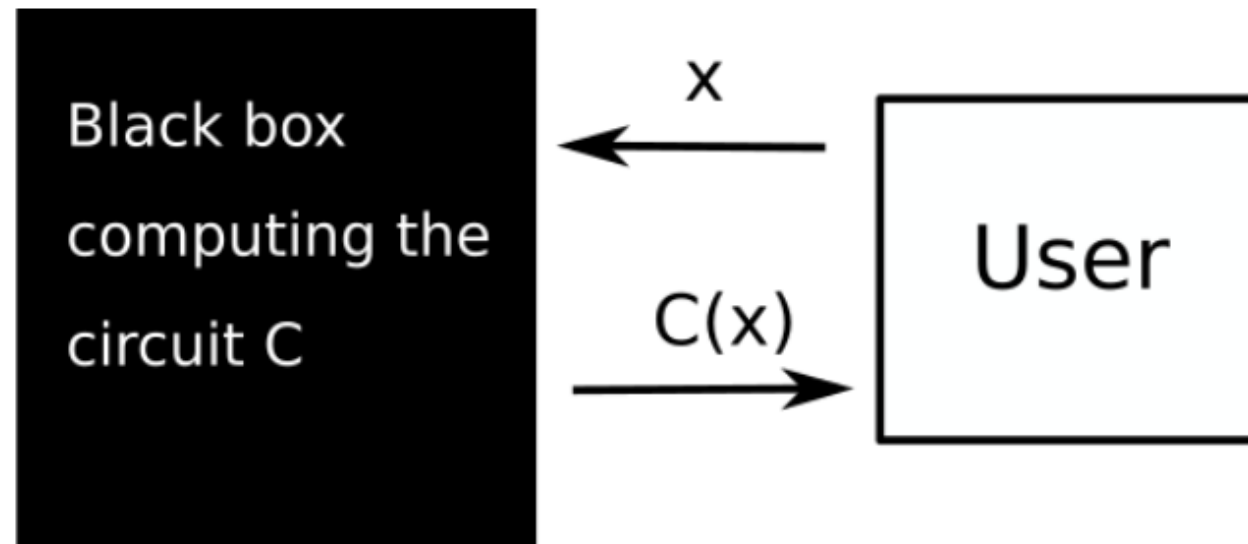


Fig. 2. Flow of information in public key system.

Wie machen wir das?

# IDEA: VIRTUAL BLACK BOX OBFUSCATOR



# IDEA: VIRTUAL BLACK BOX OBFUSCATOR



Existiert leider nicht!

```
1 def encrypt(m):  
2     sk = 0111011011011010011;  
3     c = Enc(m,sk);  
4     return c;
```

Program before obfuscation

```
1 def encrypt_after_obfuscation(m):  
2     ksghdfjkhdsfhsfkjshd9shdjf;  
3     flkdjlfkjlsjdfkslkfjdljsdfks;  
4     sdjhfkhsjkdhkfhjskdfhjskff;  
5     c = shdjfsh87h43jjsh;  
6     return c;
```

Program after obfuscation

# PUBLIC KEY CRYPTOGRAPHY



Diffie-Hellman  
key exchange

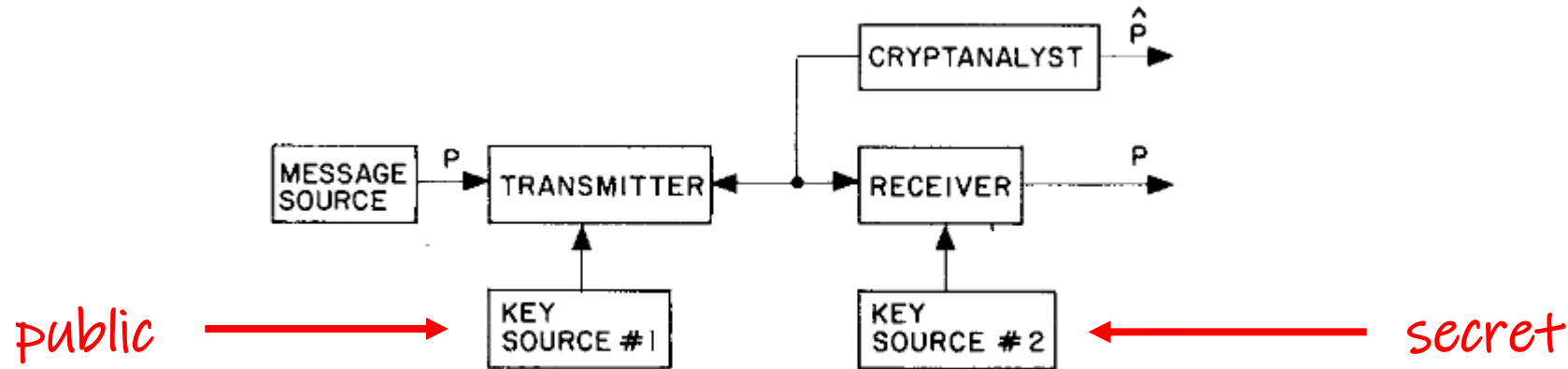


Fig. 2. Flow of information in public key system.

Each user generates an independent random number  $X_i$  chosen uniformly from the set of integers  $\{1, 2, \dots, q-1\}$ . Each keeps  $X_i$  secret, but places

public  $\longrightarrow$   $Y_i = \alpha^{X_i} \bmod q$   $\longleftarrow$  secret

in a public file with his name and address. When users  $i$  and  $j$  wish to communicate privately, they use

$$K_{ij} = \alpha^{X_i X_j} \bmod q \quad (8)$$

as their key. User  $i$  obtains  $K_{ij}$  by obtaining  $Y_j$  from the public file and letting

$$K_{ij} = Y_j^{X_i} \bmod q \quad (9)$$

$$= (\alpha^{X_j})^{X_i} \bmod q \quad (10)$$

$$= \alpha^{X_j X_i} = \alpha^{X_i X_j} \bmod q. \quad (11)$$

User  $j$  obtains  $K_{ij}$  in the similar fashion

$$K_{ij} = Y_i^{X_j} \bmod q. \quad (12)$$

# PROBLEM DES DISKRETEN LOGARITHMUS



For a cyclic group  $G$  with generator  $\omega$  every element  $\alpha \in G$  can be written as

$$\alpha = \omega^k$$

Discrete Logarithm  
Problem

Discrete Logarithm

Given  $G$ ,  $\omega$  and  $\alpha$ , compute  $k$ .

# DIE EINHEITENGRUPPE EINES ENDLICHEN KÖRPERS: MODP



Hochschule für  
Wirtschaft und Recht Berlin  
Berlin School of Economics and Law

Network Working Group  
Request for Comments: 3526  
Category: Standards Track

T. Kivinen  
M. Kojo  
SSH Communications Security  
May 2003

More Modular Exponential (MODP) Diffie-Hellman groups  
for Internet Key Exchange (IKE)

## Status of this Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

## Copyright Notice

Copyright (C) The Internet Society (2003). All Rights Reserved.



# MODP



## 3. 2048-bit MODP Group

This group is assigned id 14.

This prime is:  $2^{2048} - 2^{1984} - 1 + 2^{64} * \{ [2^{1918} \text{ pi}] + 124476 \}$

Its hexadecimal value is:

```
FFFFFFFF FFFFFFFF C90FDAA2 2168C234 C4C6628B 80DC1CD1
29024E08 8A67CC74 020BBEA6 3B139B22 514A0879 8E3404DD
EF9519B3 CD3A431B 302B0A6D F25F1437 4FE1356D 6D51C245
E485B576 625E7EC6 F44C42E9 A637ED6B 0BFF5CB6 F406B7ED
EE386BFB 5A899FA5 AE9F2411 7C4B1FE6 49286651 ECE45B3D
C2007CB8 A163BF05 98DA4836 1C55D39A 69163FA8 FD24CF5F
83655D23 DCA3AD96 1C62F356 208552BB 9ED52907 7096966D
670C354E 4ABC9804 F1746C08 CA18217C 32905E46 2E36CE3B
E39E772C 180E8603 9B2783A2 EC07A28F B5C55DF0 6F4C52C9
DE2BCBF6 95581718 3995497C EA956AE5 15D22618 98FA0510
15728E5A 8AAACAA6 FFFFFFFF FFFFFFFF
```

The generator is: 2.

## 4. 3072-bit MODP Group

This group is assigned id 15.

This prime is:  $2^{3072} - 2^{3008} - 1 + 2^{64} * \{ [2^{2942} \text{ pi}] + 1690314 \}$

Its hexadecimal value is:

```
FFFFFFFF FFFFFFFF C90FDAA2 2168C234 C4C6628B 80DC1CD1
29024E08 8A67CC74 020BBEA6 3B139B22 514A0879 8E3404DD
EF9519B3 CD3A431B 302B0A6D F25F1437 4FE1356D 6D51C245
E485B576 625E7EC6 F44C42E9 A637ED6B 0BFF5CB6 F406B7ED
EE386BFB 5A899FA5 AE9F2411 7C4B1FE6 49286651 ECE45B3D
C2007CB8 A163BF05 98DA4836 1C55D39A 69163FA8 FD24CF5F
83655D23 DCA3AD96 1C62F356 208552BB 9ED52907 7096966D
670C354E 4ABC9804 F1746C08 CA18217C 32905E46 2E36CE3B
E39E772C 180E8603 9B2783A2 EC07A28F B5C55DF0 6F4C52C9
DE2BCBF6 95581718 3995497C EA956AE5 15D22618 98FA0510
15728E5A 8AAAC42D AD33170D 04507A33 A85521AB DF1CBA64
ECFB8504 58DBEF0A 8AEA7157 5D060C7D B3970F85 A6E1E4C7
ABF5AE8C DB0933D7 1E8C94E0 4A25619D CEE3D226 1AD2EE6B
F12FFA06 D98A0864 D8760273 3EC86A64 521F2B18 177B200C
BBE11757 7A615D6C 770988C0 BAD946E2 08E24FA0 74E5AB31
43DB5BFC E0FD108E 4B82D120 A93AD2CA FFFFFFFF FFFFFFFF
```

The generator is: 2.

## 5. 4096-bit MODP Group

This group is assigned id 16.

This prime is:  $2^{4096} - 2^{4032} - 1 + 2^{64} * \{ [2^{3966} \text{ pi}] + 240904 \}$

Its hexadecimal value is:

```
FFFFFFFF FFFFFFFF C90FDAA2 2168C234 C4C6628B 80DC1CD1
29024E08 8A67CC74 020BBEA6 3B139B22 514A0879 8E3404DD
EF9519B3 CD3A431B 302B0A6D F25F1437 4FE1356D 6D51C245
E485B576 625E7EC6 F44C42E9 A637ED6B 0BFF5CB6 F406B7ED
EE386BFB 5A899FA5 AE9F2411 7C4B1FE6 49286651 ECE45B3D
C2007CB8 A163BF05 98DA4836 1C55D39A 69163FA8 FD24CF5F
83655D23 DCA3AD96 1C62F356 208552BB 9ED52907 7096966D
670C354E 4ABC9804 F1746C08 CA18217C 32905E46 2E36CE3B
E39E772C 180E8603 9B2783A2 EC07A28F B5C55DF0 6F4C52C9
DE2BCBF6 95581718 3995497C EA956AE5 15D22618 98FA0510
15728E5A 8AAAC42D AD33170D 04507A33 A85521AB DF1CBA64
ECFB8504 58DBEF0A 8AEA7157 5D060C7D B3970F85 A6E1E4C7
ABF5AE8C DB0933D7 1E8C94E0 4A25619D CEE3D226 1AD2EE6B
F12FFA06 D98A0864 D8760273 3EC86A64 521F2B18 177B200C
BBE11757 7A615D6C 770988C0 BAD946E2 08E24FA0 74E5AB31
43DB5BFC E0FD108E 4B82D120 A93AD2CA FFFFFFFF FFFFFFFF
```

The generator is: 2.

# MODP



## 8. Security Considerations

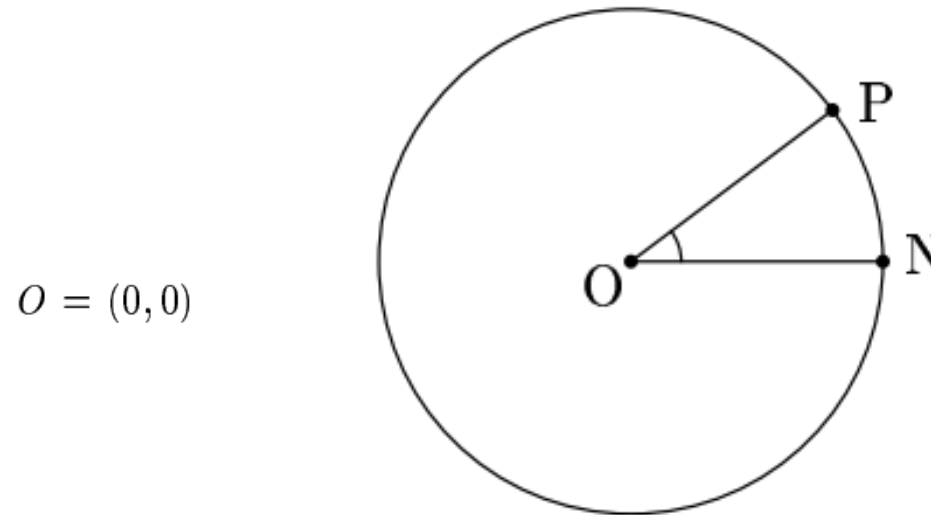
This document describes new stronger groups to be used in IKE. The strengths of the groups defined here are always estimates and there are as many methods to estimate them as there are cryptographers. For the strength estimates below we took the both ends of the scale so the actual strength estimate is likely between the two numbers given here.

Group	Modulus	Strength Estimate 1		Strength Estimate 2	
		exponent		exponent	
		in bits	size	in bits	size
5	1536-bit	90	180-	120	240-
14	2048-bit	110	220-	160	320-
15	3072-bit	130	260-	210	420-
16	4096-bit	150	300-	240	480-
17	6144-bit	170	340-	270	540-
18	8192-bit	190	380-	310	620-

# BEISPIEL: GRUPPE AUF EINEM KREIS



$$\lambda : \mathbb{R} \longrightarrow \mathcal{C}(\mathbb{R}) : \alpha \longmapsto (\cos 2\pi\alpha, \sin 2\pi\alpha)$$



$$O = (0, 0)$$

$$N = (1, 0)$$

Gruppengesetz  
(funktioniert über  
jedem Ring)

$$\begin{aligned}\cos 2\pi(\alpha_1 + \alpha_2) &= \cos 2\pi\alpha_1 \cos 2\pi\alpha_2 - \sin 2\pi\alpha_1 \sin 2\pi\alpha_2, \\ \sin 2\pi(\alpha_1 + \alpha_2) &= \cos 2\pi\alpha_1 \sin 2\pi\alpha_2 + \cos 2\pi\alpha_2 \sin 2\pi\alpha_1,\end{aligned}$$

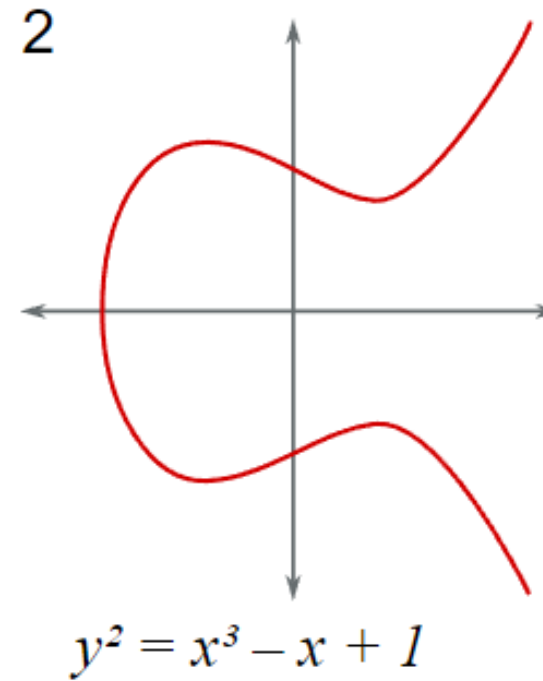
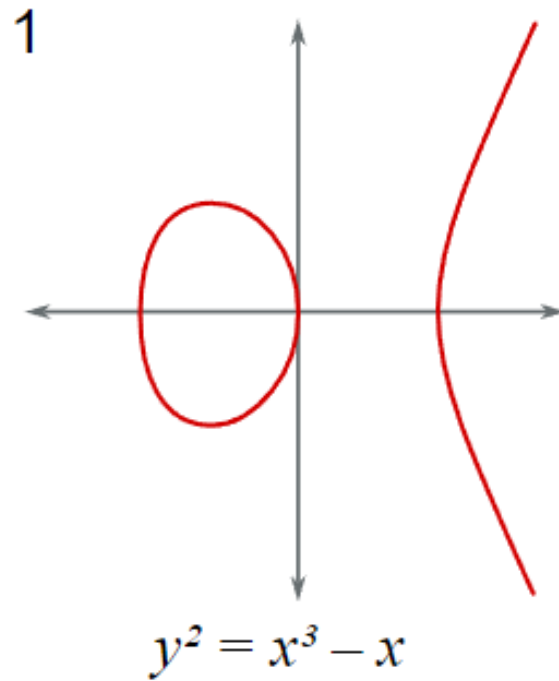
$$(x_1, y_1) + (x_2, y_2) = (x_1 x_2 - y_1 y_2, x_1 y_2 + x_2 y_1).$$

$$-(x, y) = (x, -y)$$

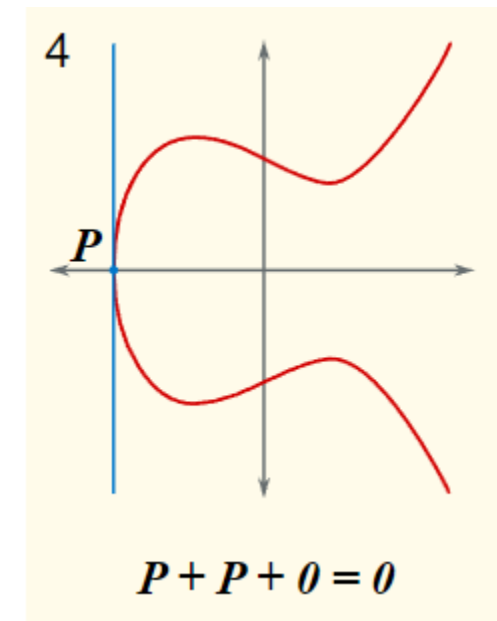
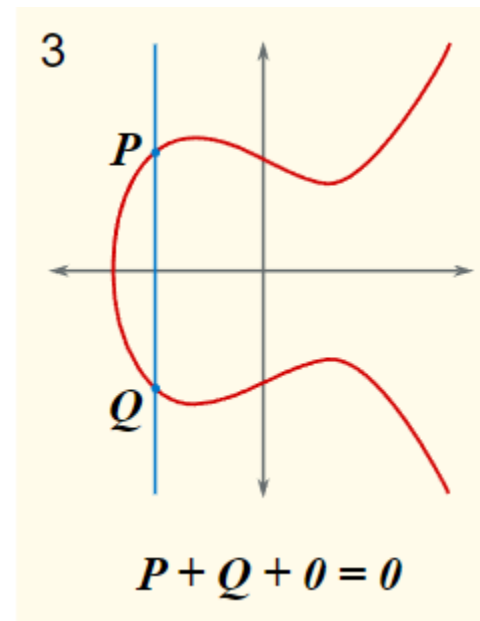
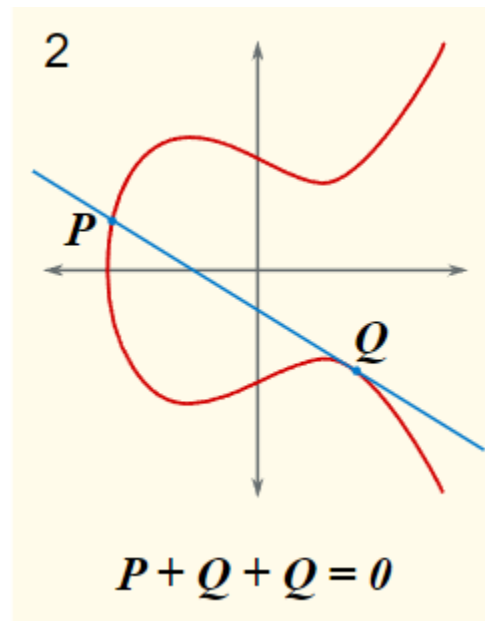
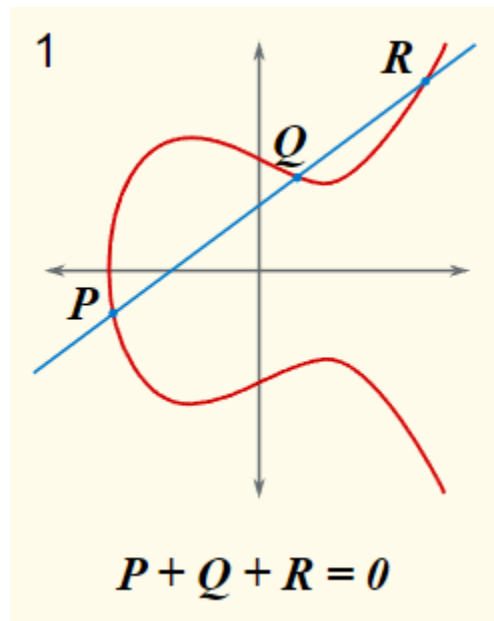
Neutrales Element

Inverses Element

# ELLIPTISCHE KURVEN



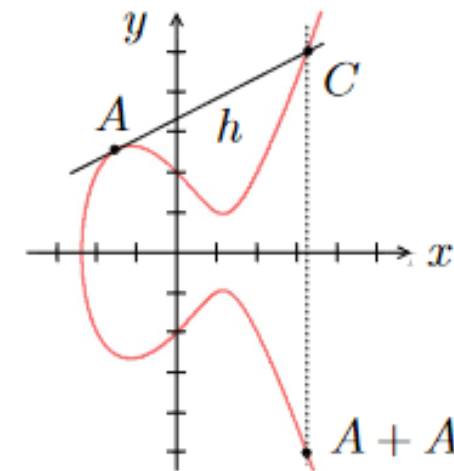
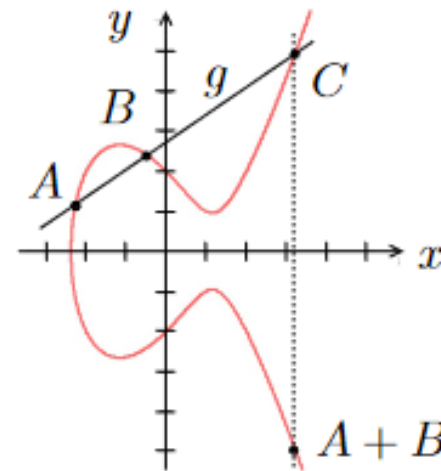
# ELLIPTISCHE KURVEN



# ELLIPTISCHE KURVEN



elliptic curve  $E$  given by the equation  $y^2 = x^3 - 4x + 4$



*Definition.* We define  $+: E \times E \rightarrow E, (A, B) \mapsto A + B$  as follows.

- (a) We set  $A + O = O + A := O$  for all  $A$ .
- (b) If  $(x_A, y_A) = (x_B, -y_B)$ , then  $A + B := O$ .
- (c) If  $(x_A, y_A) \neq (x_B, -y_B)$ , then we define  $A + B := (x_{AB}, y_{AB})$  where

$$(1) \quad \begin{aligned} x_{AB} &:= \alpha(A, B)^2 - x_A - x_B \\ y_{AB} &:= -y_A + \alpha(A, B)(x_A - x_{AB}) \end{aligned}$$

with  $\alpha(A, B) = \frac{y_A - y_B}{x_A - x_B}$  if  $x_A \neq x_B$  and  $\alpha(A, B) = \frac{3x_A^2 + a}{2y_A}$  if  $x_A = x_B$ .

# ELLIPTISCHE KURVEN



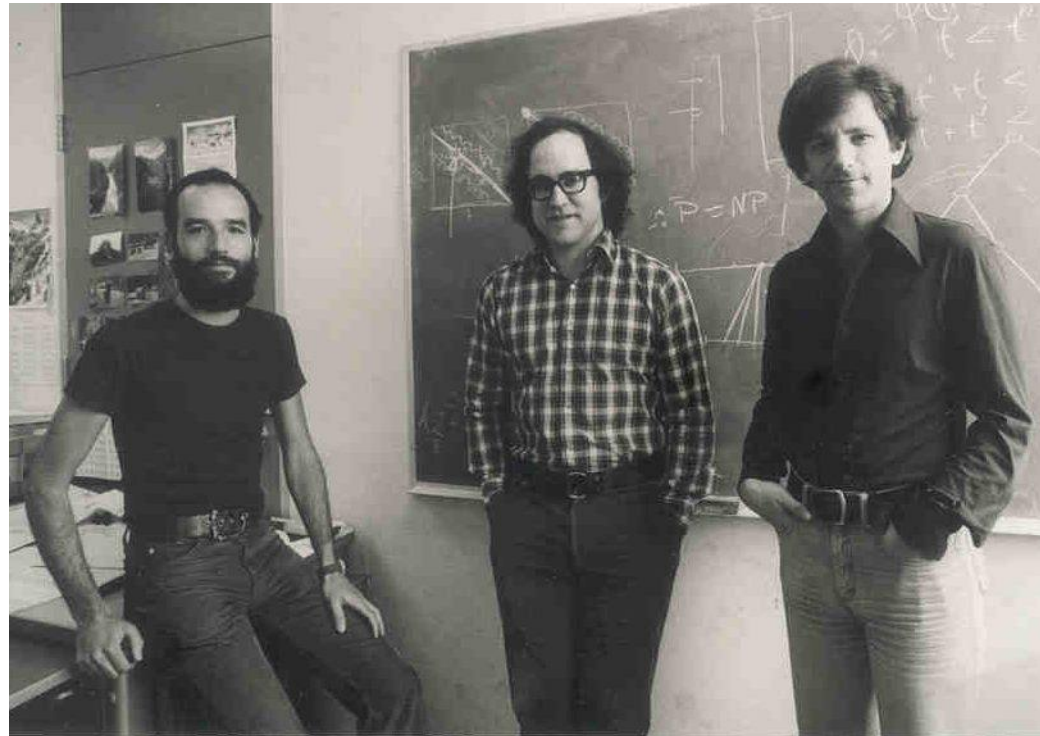
		Parameters:			ECDLP security:				ECC security:			
Curve	Safe?	field	equation	base	rho	transfer	disc	rigid	ladder	twist	complete	ind
Anomalous	False	True✓	True✓	True✓	True✓	False	False	True✓	False	False	False	False
M-221	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓
E-222	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓
NIST P-224	False	True✓	True✓	True✓	True✓	True✓	True✓	False	False	False	False	False
Curve1174	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓
Curve25519	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓
BN(2,254)	False	True✓	True✓	True✓	True✓	False	False	True✓	False	False	False	False
brainpoolP256t1	False	True✓	True✓	True✓	True✓	True✓	True✓	True✓	False	False	False	False
ANSSI FRP256v1	False	True✓	True✓	True✓	True✓	True✓	True✓	False	False	False	False	False
NIST P-256	False	True✓	True✓	True✓	True✓	True✓	True✓	False	False	True✓	False	False
secp256k1	False	True✓	True✓	True✓	True✓	True✓	False	True✓	False	True✓	False	False
E-382	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓
M-383	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓
Curve383187	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓
brainpoolP384t1	False	True✓	True✓	True✓	True✓	True✓	True✓	True✓	False	True✓	False	False
NIST P-384	False	True✓	True✓	True✓	True✓	True✓	True✓	False	False	True✓	False	False
Curve41417	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓
Ed448-Goldilocks	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓
M-511	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓
E-521	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓



# RIVEST SHAMIR ADLEMAN (1977)



Hochschule für  
Wirtschaft und Recht Berlin  
Berlin School of Economics and Law



## A Method for Obtaining Digital Signatures and Public-Key Cryptosystems

R.L. Rivest, A. Shamir, and L. Adleman\*



# RIVEST SHAMIR ADLEMAN



Hochschule für  
Wirtschaft und Recht Berlin  
Berlin School of Economics and Law

## A Method for Obtaining Digital Signatures and Public-Key Cryptosystems

R.L. Rivest, A. Shamir, and L. Adleman\*

A message is encrypted by representing it as a number  $M$ , raising  $M$  to a ~~publicly specified power~~  $e$ , and then taking the remainder when the result is divided by the publicly specified product,  $n$ , of two large secret prime numbers  $p$  and  $q$ . Decryption is similar; only a different, secret, power  $d$  is used, where  $e \cdot d \equiv 1 \pmod{(p-1) \cdot (q-1)}$ . The security of the system rests in part on the difficulty of factoring the published divisor,  $n$ .

*public* →  $e$  →  $n$  →  $d$  *secret*

# RIVEST SHAMIR ADLEMAN



## RSA Algorithm

### Key Generation

Select $p, q$	$p$ and $q$ both prime; $p \neq q$
Calculate $n = p \times q$	
Calculate $\phi(n) = (p-1)(q-1)$	
Select integer $e$	$\gcd(\phi(n), e) = 1; 1 < e < \phi(n)$
Calculate $d$	$de \bmod \phi(n) = 1$
Public key	$KU = \{e, n\}$
Private key	$KR = \{d, n\}$

### Encryption

Plaintext:	$M < n$
Ciphertext:	$C = M^e \bmod n$

### Decryption

Plaintext:	$M$
Ciphertext:	$M = C^d \bmod n$

# FAKTORISIERUNGSPROBLEM



Hochschule für  
Wirtschaft und Recht Berlin  
Berlin School of Economics and Law

Given a positive integer  $n = pq$ , where  $p$  and  $q$  are primes.

Compute  $p$  and  $q$ .

# FAKTORISIERUNGSPROBLEM



## General number field sieve

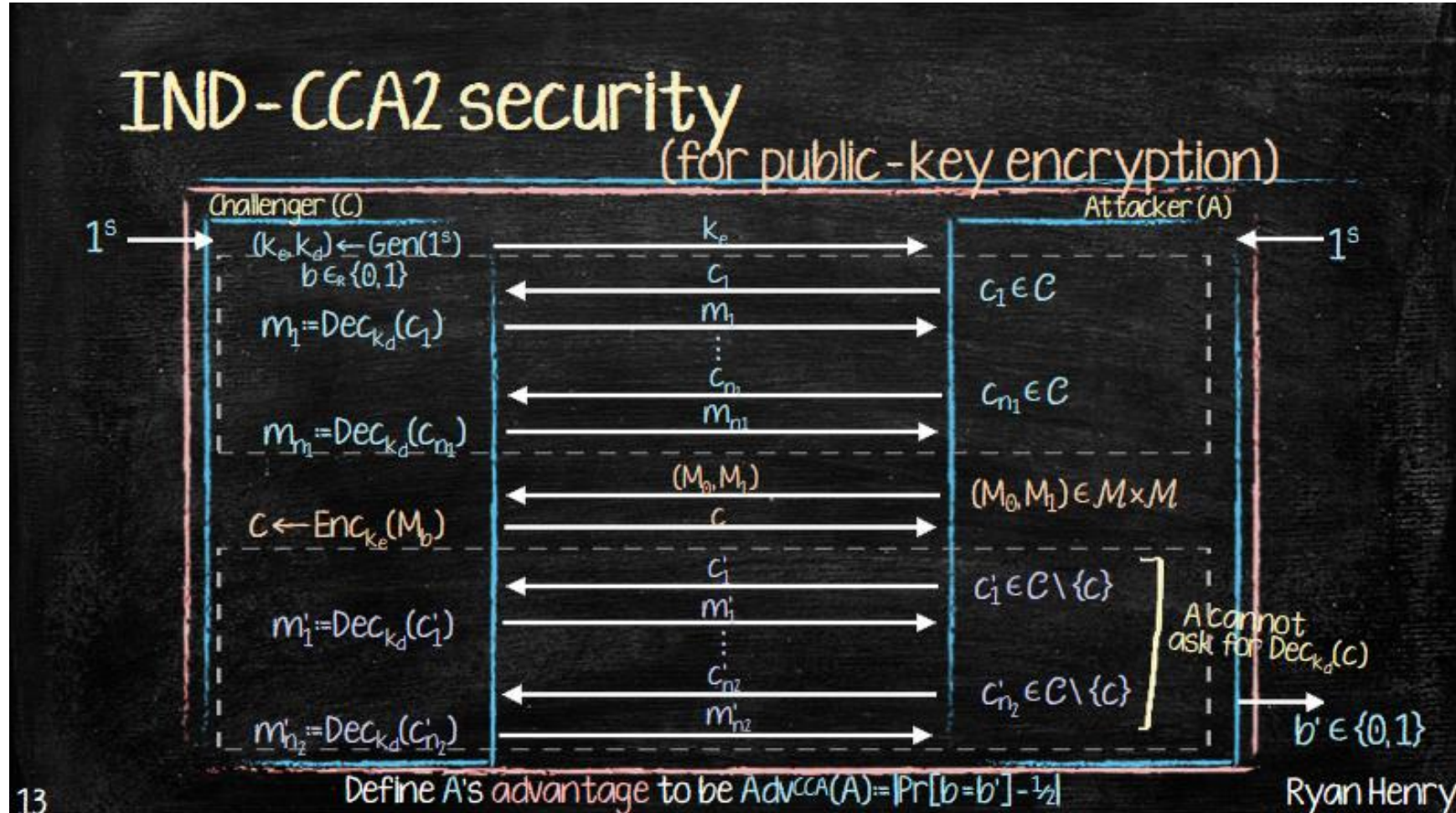
From Wikipedia, the free encyclopedia

In [number theory](#), the **general number field sieve (GNFS)** is the most [efficient classical algorithm](#) known for [factoring integers](#) larger than  $10^{100}$ . [Heuristically](#), its [complexity](#) for factoring an integer  $n$  (consisting of  $\lfloor \log_2 n \rfloor + 1$  bits) is of the form

$$\exp\left(\left(\sqrt[3]{\frac{64}{9}} + o(1)\right) (\ln n)^{\frac{1}{3}} (\ln \ln n)^{\frac{2}{3}}\right) = L_n \left[\frac{1}{3}, \sqrt[3]{\frac{64}{9}}\right]$$

Wir werden später noch  
sehen, was das genau  
bedeuten soll...

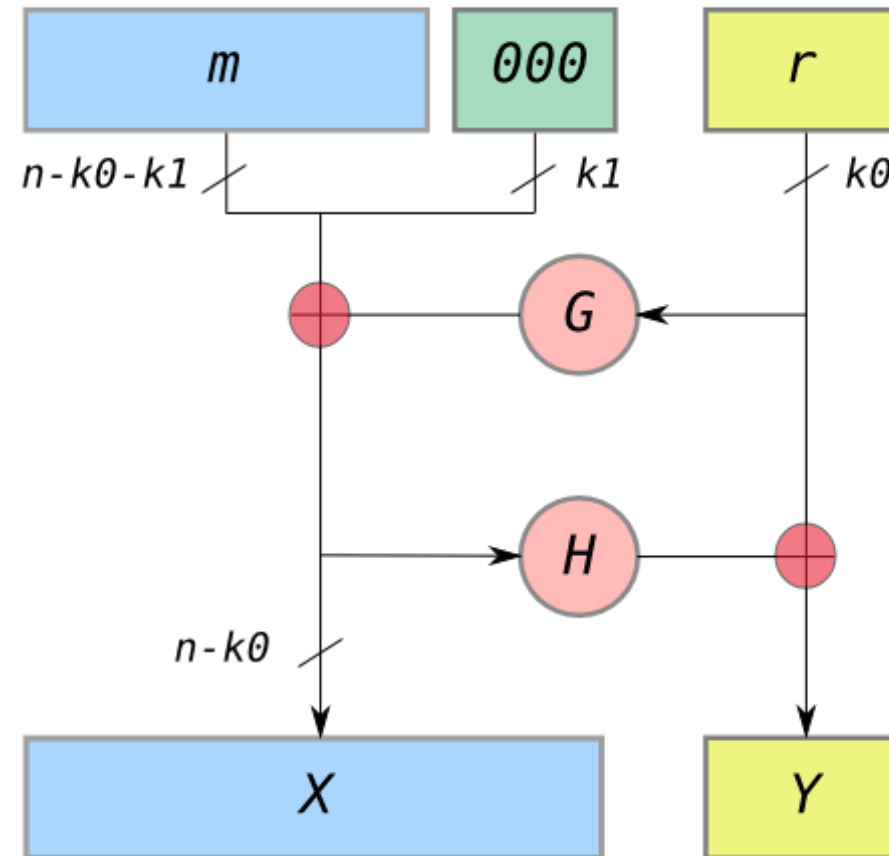
# WIE MISST MAN (EIGENTLICH) SICHERHEIT? (3. ANSATZ)



# OPTIMAL ASYMMETRIC ENCRYPTION PADDING (OAEP)



Hochschule für  
Wirtschaft und Recht Berlin  
Berlin School of Economics and Law





# ELGAMAL



## ElGamal Encryption

### ◆ Key generation

- Pick a large prime  $p$ , generator  $g$  of  $Z_p^*$
- Private key: random  $x$  such that  $1 \leq x \leq p-2$
- Public key:  $(p, g, \gamma = g^x \bmod p)$

### ◆ Encryption

- Pick random  $k$ ,  $1 \leq k \leq p-2$
- $E(m) = (g^k \bmod p, m \cdot \gamma^k \bmod p) = (\gamma, \delta)$

### ◆ Decryption

- Given ciphertext  $(\gamma, \delta)$ , compute  $\gamma^{-x} \bmod p$
- Recover  $m = \delta \cdot (\gamma^{-x}) \bmod p$

slide 7



Taher Elgamal

# ELGAMAL



---

**Algorithm** Key generation for generalized ElGamal public-key encryption

---

SUMMARY: each entity creates a public key and a corresponding private key.  
Each entity  $A$  should do the following:

1. Select an appropriate cyclic group  $G$  of order  $n$ , with generator  $\alpha$ . (It is assumed here that  $G$  is written multiplicatively.)
  2. Select a random integer  $a$ ,  $1 \leq a \leq n - 1$ , and compute the group element  $\alpha^a$ .
  3.  $A$ 's public key is  $(\alpha, \alpha^a)$ , together with a description of how to multiply elements in  $G$ ;  $A$ 's private key is  $a$ .
- 

---

**Algorithm** Generalized ElGamal public-key encryption

---

SUMMARY:  $B$  encrypts a message  $m$  for  $A$ , which  $A$  decrypts.

1. *Encryption.*  $B$  should do the following:
    - (a) Obtain  $A$ 's authentic public key  $(\alpha, \alpha^a)$ .
    - (b) Represent the message as an element  $m$  of the group  $G$ .
    - (c) Select a random integer  $k$ ,  $1 \leq k \leq n - 1$ .
    - (d) Compute  $\gamma = \alpha^k$  and  $\delta = m \cdot (\alpha^a)^k$ .
    - (e) Send the ciphertext  $c = (\gamma, \delta)$  to  $A$ .
  2. *Decryption.* To recover plaintext  $m$  from  $c$ ,  $A$  should do the following:
    - (a) Use the private key  $a$  to compute  $\gamma^a$  and then compute  $\gamma^{-a}$ .
    - (b) Recover  $m$  by computing  $(\gamma^{-a}) \cdot \delta$ .
-