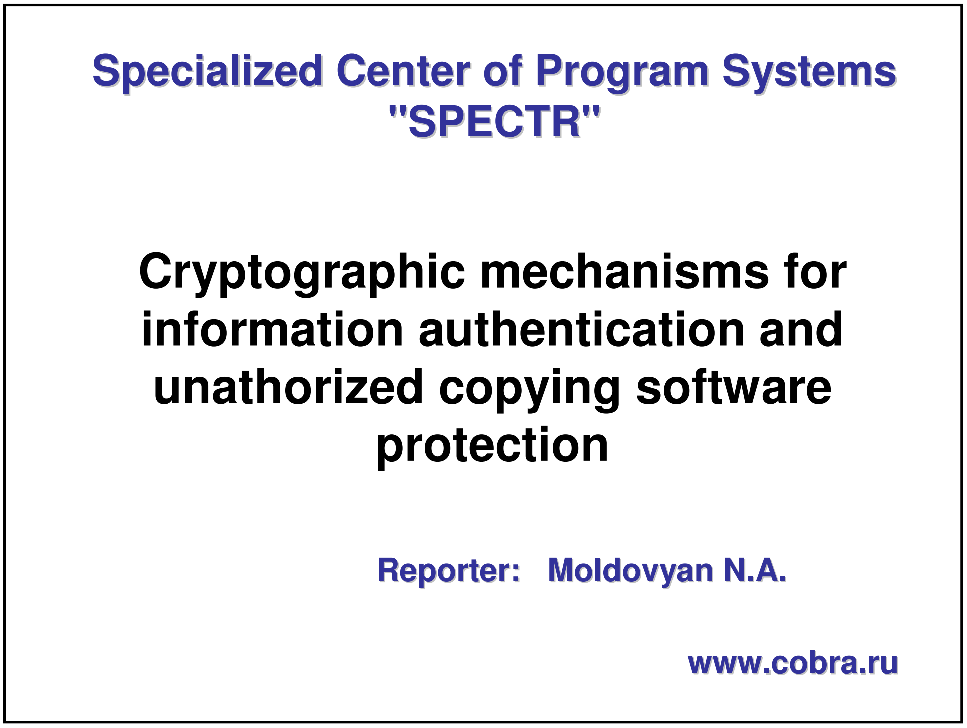
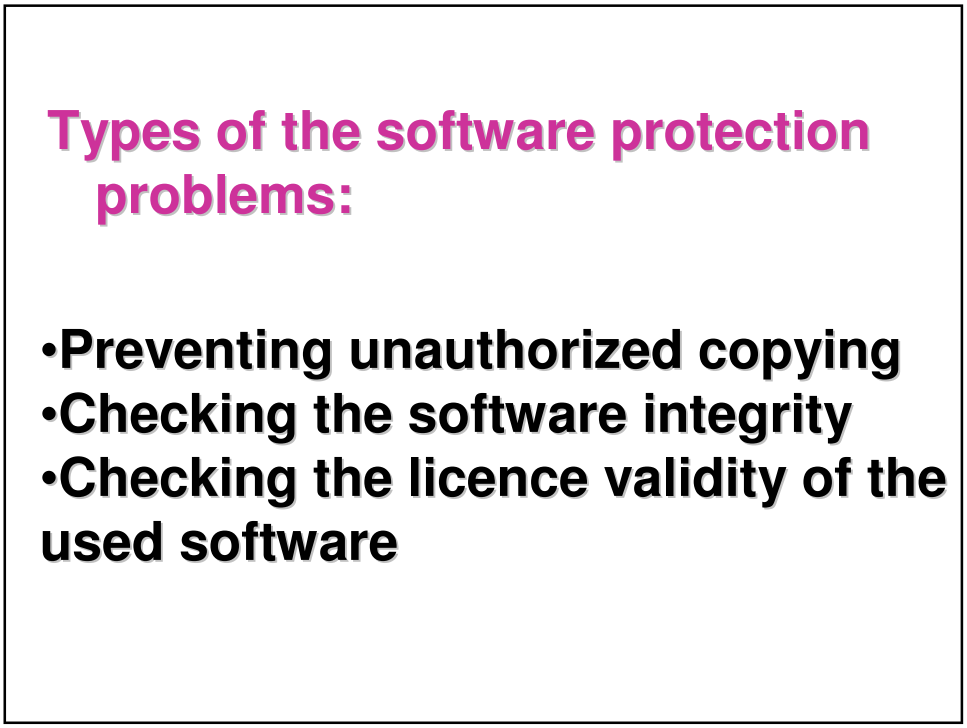


Specialized Center of Program Systems "SPECTR"

Cryptographic mechanisms for information authentication and unathorized copying software protection

Reporter: Moldovyan N.A.

www.cobra.ru



Types of the software protection problems:

- Preventing unauthorized copying
- Checking the software integrity
- Checking the licence validity of the used software

Three cryptographic mechanism used in solving the mentioned problems

- Data ciphering using symmetric or asymmetric encryption algorithms
- Computation of the cryptographic checksums
- Information authentication with public key digital signatures

Mechanisms for preventing unauthorized copying

- Decryption of the current portions of the executed program just before they are moved in RAM
- Decryption of the current portions of the executed program just before they are moved in microprocessor registers
- Decryption with the CPU
- Decryption with some external device (for example, USB-device)

These mechanisms needs fast software suitable block ciphers and/or fast and cheap in hardware encryption algorithms

Mechanisms for preventing unauthorized modification of the software

- Computing and checking the hash values of the software before execution
- Dynamic computing of the hash values from each current portion of the software
- Computing and checking the hash values combined with checking the digital signatures corresponding to the hash values

These mechanisms needs for fast hash functions and efficient digital signature algorithms

Mechanisms for checking the licence validity of the used software

- Using the digital signatures to electronic versions of the licences
- Using the digital signatures on the hard copyes of the licences, which are computed from the message representing the concatenation of the licence agreement text and scanned special "hardware" label on the licence
- Using the digital water marking (steganography)

These mechanisms needs for fast and secure digital signature algorithms

Directions of our research in applied cryptography - I

In relation to the indicated ways and directions of the use of cryptographic methods for the software protection our research concerns the following areas:

- Design of the fast software oriented block ciphers (the main mechanism is the data-dependent subkey selection)
- Design of the fast block ciphers suitable to very cheap implementation (the main used primitives are data-driven operations performed with the permutation and substitution-permutation networks)

Directions of our research in applied cryptography - II

- Design of the digital signature algorithms providing short signatures while using different hard computational problems
- Design of the provably secure digital signatures and public key encryption algorithms
- Design of computationally efficient digital signature algorithms (based on new computational problems and new algebraic structures)

Results on block cipher design - I

1. Different types of the data-driven operations have been introduced and used while designing fast block ciphers: data-dependent subkey selection, variable bit permutations, switchable data-dependent operations.
2. Controlled elements for designing the substitution-permutation networks have been classified.
3. A set of fast block ciphers suitable for software and hardware implementations have been designed.
4. A universal architecture for bit permutation instruction have been proposed

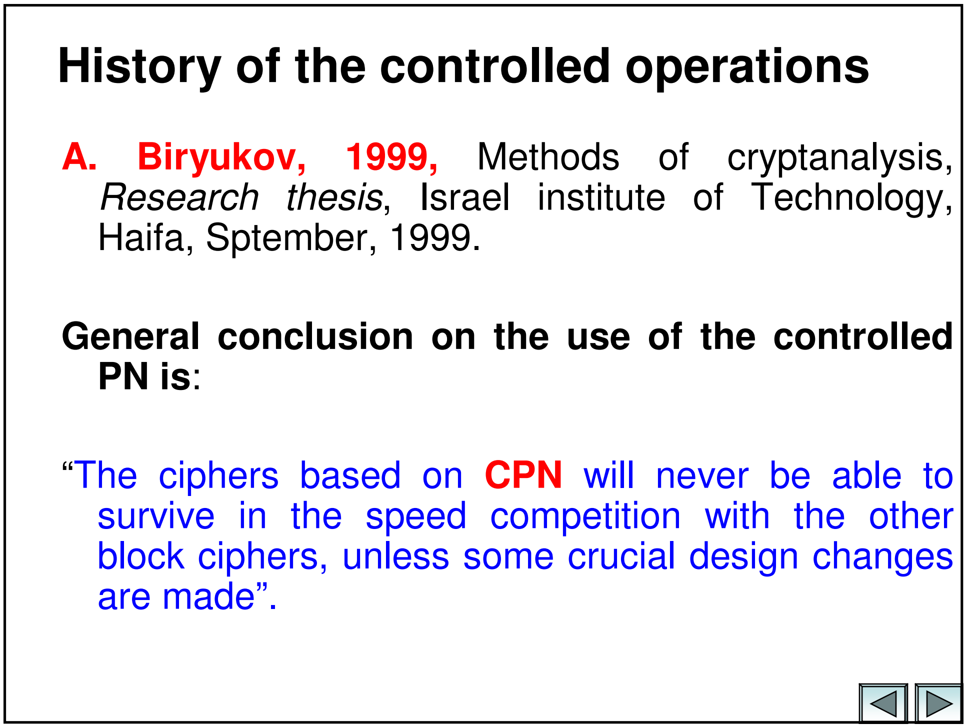
History of the controlled operations

J.B. Kam, G.I. Davida, 1979, Structured Design of Substitution-Permutation Encryption Networks, *IEEE Transactions on Computers*, vol. C-28 (1979), pp. 747-753

M. Portz, 1992, A generallized description of DES-based and Benes-based permutation generators, *Lect. Notes Comput. Sci.*, Berlin:Springer-Verlag, vol. 718 (1992) pp. 397-409.

M.Kwan, 1997, The design of the ICE encryption algorithm, Proceedings of the 4th International Workshop, Fast Software Encryption - FSE '97, *Lecture Notes in Computer Science*, Berlin: Springer-Verlag, vol. 1267 (1997) pp. 69-82.

A

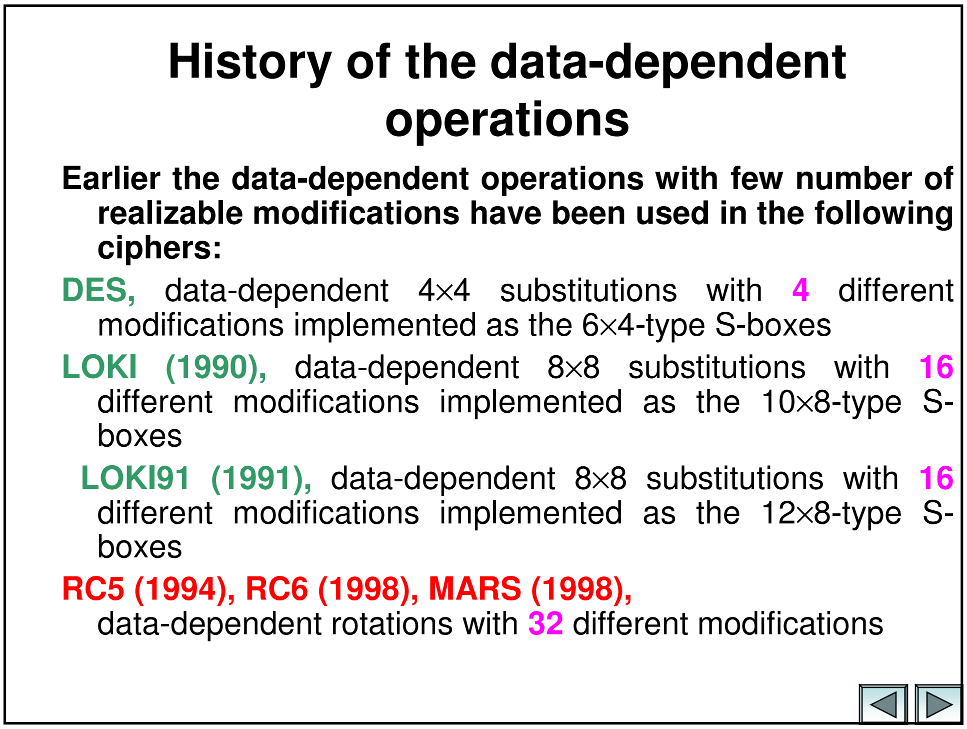


History of the controlled operations

A. Biryukov, 1999, Methods of cryptanalysis, *Research thesis*, Israel institute of Technology, Haifa, Sptember, 1999.

General conclusion on the use of the controlled PN is:

“The ciphers based on **CPN** will never be able to survive in the speed competition with the other block ciphers, unless some crucial design changes are made”.



History of the data-dependent operations

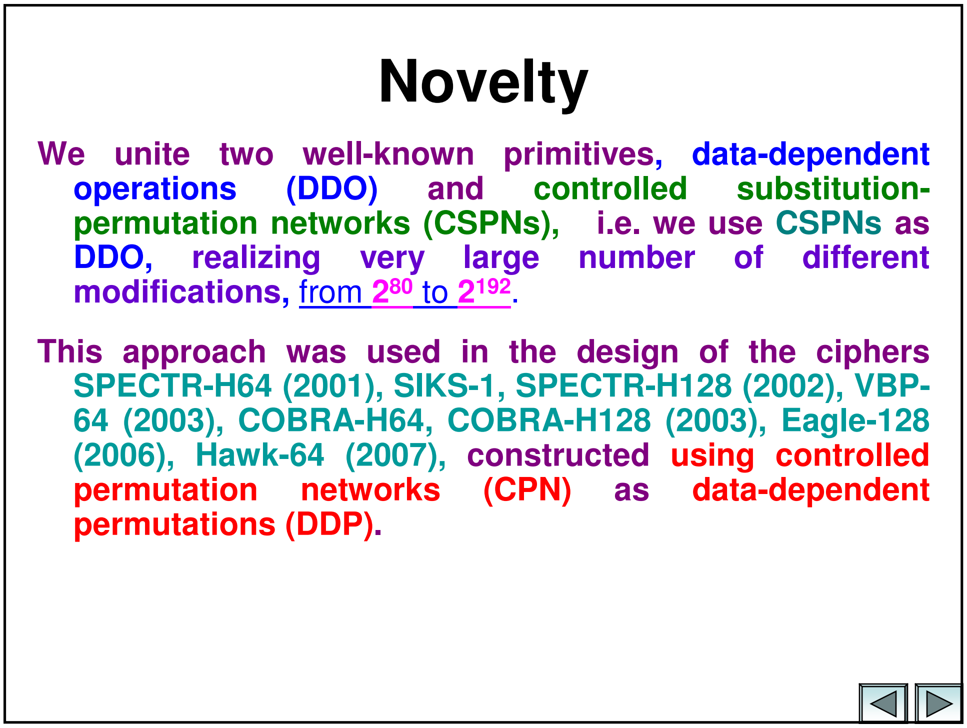
Earlier the data-dependent operations with few number of realizable modifications have been used in the following ciphers:

DES, data-dependent 4×4 substitutions with **4** different modifications implemented as the 6×4-type S-boxes

LOKI (1990), data-dependent 8×8 substitutions with **16** different modifications implemented as the 10×8-type S-boxes

LOKI91 (1991), data-dependent 8×8 substitutions with **16** different modifications implemented as the 12×8-type S-boxes

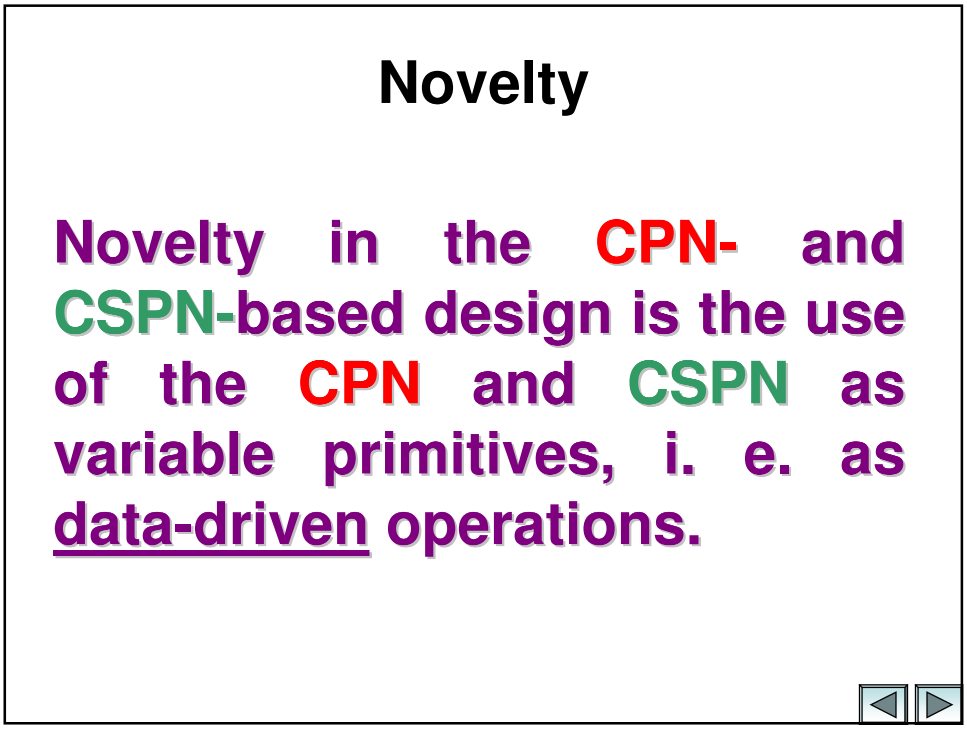
RC5 (1994), RC6 (1998), MARS (1998), data-dependent rotations with **32** different modifications



Novelty

We unite two well-known primitives, data-dependent operations (DDO) and controlled substitution-permutation networks (CSPNs), i.e. we use CSPNs as DDO, realizing very large number of different modifications, from 2^{80} to 2^{192}.

This approach was used in the design of the ciphers SPECTR-H64 (2001), SIKS-1, SPECTR-H128 (2002), VBP-64 (2003), COBRA-H64, COBRA-H128 (2003), Eagle-128 (2006), Hawk-64 (2007), constructed using controlled permutation networks (CPN) as data-dependent permutations (DDP).



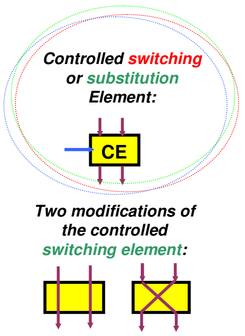
Novelty

Novelty in the CPN- and CSPN-based design is the use of the CPN and CSPN as variable primitives, i. e. as data-driven operations.

Designing primitives

Below the construction of the operational boxes for implementing the data-driven operations is considered.

CPN-LIKE SUBSTITUTION-PERMUTATION NETWORKS

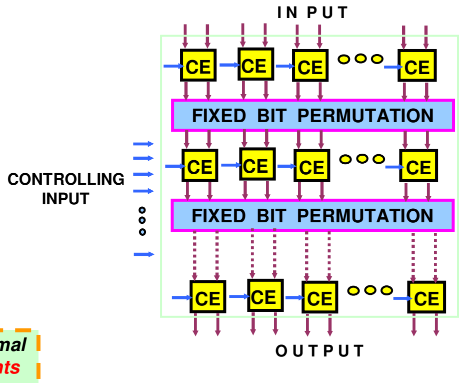


Controlled switching or substitution Element:

CE

Two modifications of the controlled switching element:

There exist 288 nonlinear minimal controlled substitution elements



IN PUT

CE CE CE CE

FIXED BIT PERMUTATION

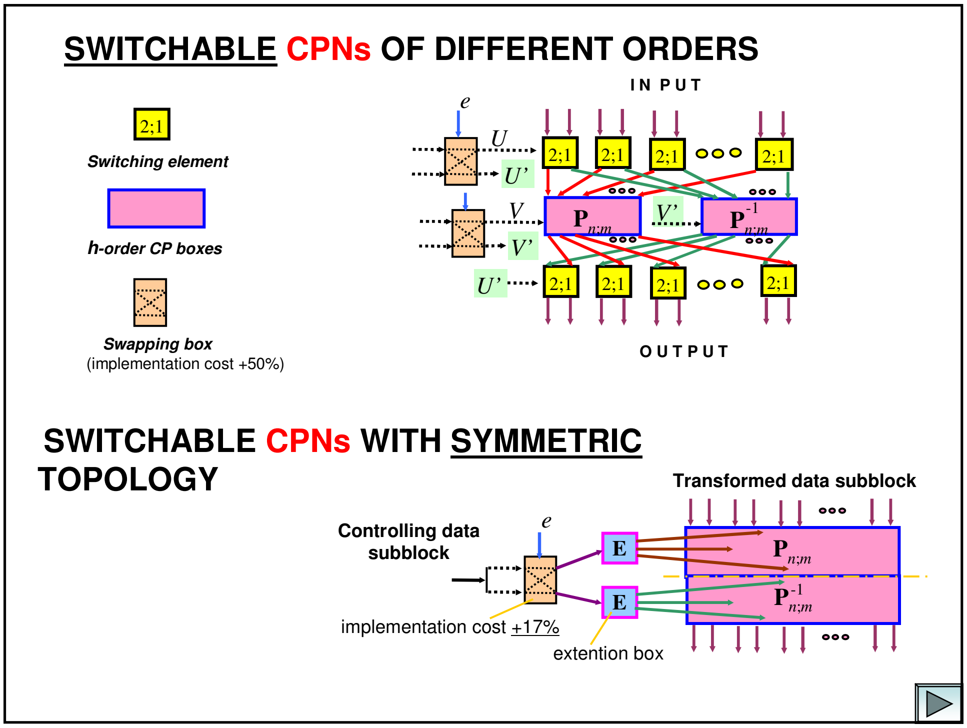
CONTROLLING INPUT

CE CE CE CE

FIXED BIT PERMUTATION

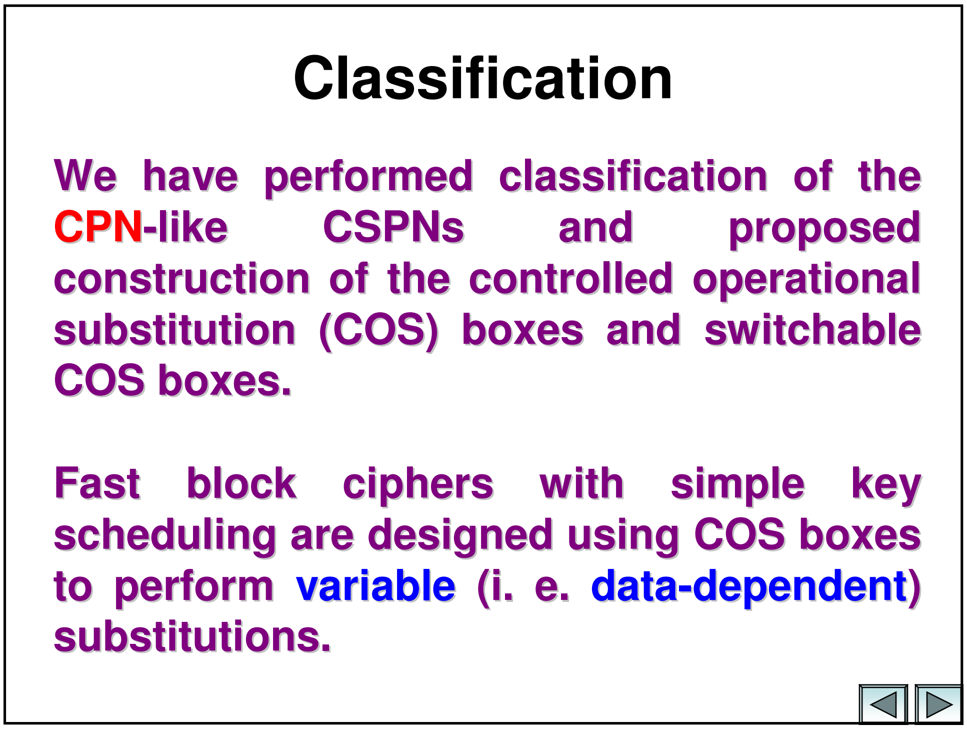
CE CE CE CE

OUTPUT



SWITCHABLE CPNs OF DIFFERENT ORDERS

SWITCHABLE CPNs WITH SYMMETRIC TOPOLOGY

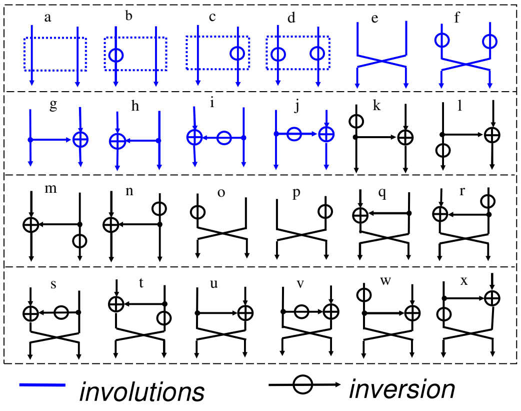


Classification

We have performed classification of the CPN-like CSPNs and proposed construction of the controlled operational substitution (COS) boxes and switchable COS boxes.

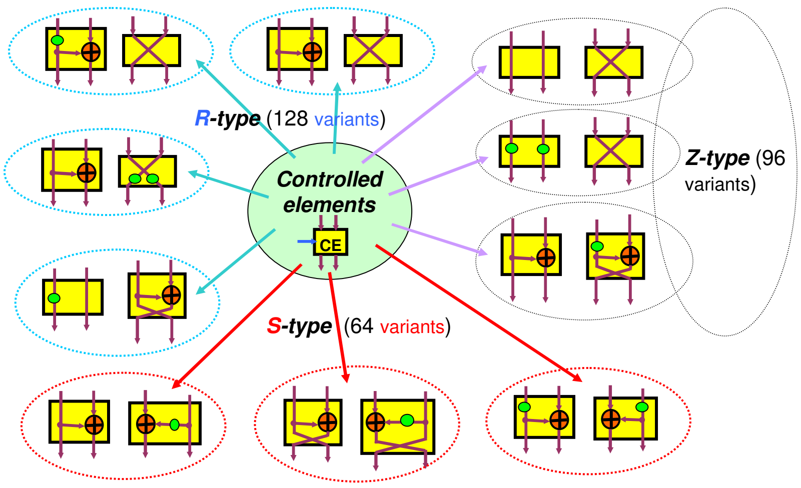
Fast block ciphers with simple key scheduling are designed using COS boxes to perform variable (i. e. data-dependent) substitutions.

ALL EXISTING 2×2-TYPE SUBSTITUTIONS

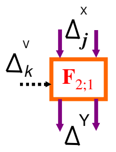


	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x
a					P	Z									Z	Z	R	R	R	R	R	R	R	R
b					Z	Z									Z	Z	R	R	R	R	R	R	R	R
c					Z	Z									Z	Z	R	R	R	R	R	R	R	R
d					Z	Z									Z	Z	R	R	R	R	R	R	R	R
e	P	Z	Z	Z			R	R	R	R	R	R	R	R										
f	Z	Z	Z	Z			R	R	R	R	R	R	R	R										
g					R	R		S	S				S	S	R	R					Z	Z	Z	Z
h					R	R	S			S	S	S			R	R	Z	Z	Z	Z				
i					R	R	S			S	S	S			R	R	Z	Z	Z	Z				
j					R	R		S	S				S	S	R	R					Z	Z	Z	Z
k					R	R		S	S				S	S	R	R					Z	Z	Z	Z
l					R	R		S	S				S	S	R	R					Z	Z	Z	Z
m					R	R	S			S	S	S			R	R	Z	Z	Z	Z				
n					R	R	S			S	S	S			R	R	Z	Z	Z	Z				
o	Z	Z	Z	Z			R	R	R	R	R	R	R	R										
p	Z	Z	Z	Z			R	R	R	R	R	R	R	R										
q	R	R	R	R				Z	Z				Z	Z							S	S	S	S
r	R	R	R	R				Z	Z				Z	Z							S	S	S	S
s	R	R	R	R				Z	Z				Z	Z							S	S	S	S
t	R	R	R	R				Z	Z				Z	Z							S	S	S	S
u	R	R	R	R			Z			Z	Z	Z					S	S	S	S				
v	R	R	R	R			Z			Z	Z	Z					S	S	S	S				
w	R	R	R	R			Z			Z	Z	Z					S	S	S	S				
x	R	R	R	R			Z			Z	Z	Z					S	S	S	S				

MAIN TYPES OF THE BUILDING ELEMENTS OF THE CPN-LIKE CSPNs



DIFFERENTIAL PROPERTIES OF THE R-, S- AND Z-TYPE ELEMENTS

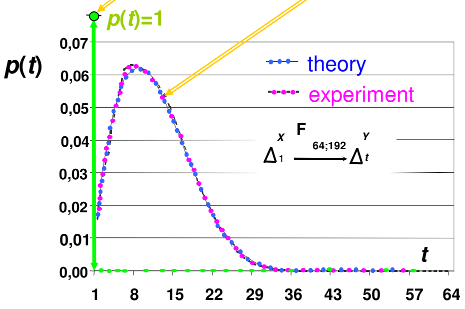


Pr $(\Delta_i^Y / \overset{x}{\Delta}_j, \Delta_k^V)$

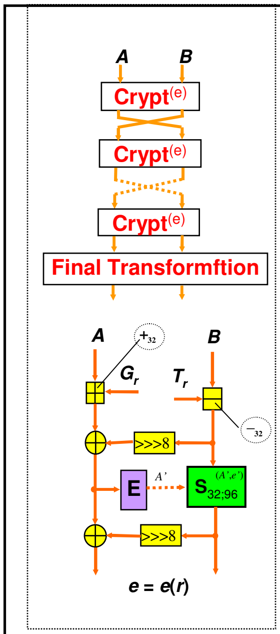
$k = 0, 1; \quad i, j = 0, 1, 2$

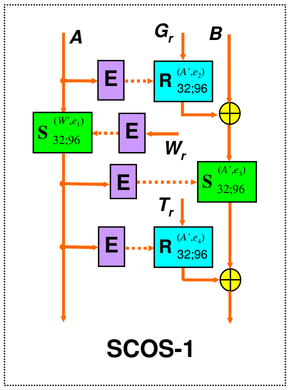
i	*j*	*k*	S	R	Z	Z	Z	Z ($P_{2;1}$)
0	0	1	1/4	1/4	0	1/2	0	1/2
1	0	1	1/2	1/2	1	0	1	0
2	0	1	1/4	1/4	0	1/2	0	1/2
0	1	1	1/4	1/4	1/4	1/4	1/2	0
1	1	1	1/2	1/2	1/2	1/2	0	1
2	1	1	1/4	1/4	1/4	1/4	1/2	0
1	1	0	1/2	3/4	1/2	1/2	1	1
2	1	0	1/2	1/4	1/2	1/2	0	0
1	2	0	1	1/2	1	1	0	0
2	2	0	0	1/2	0	0	1	0
0	2	1	1/4	1/4	1/2	0	0	1/2
1	2	1	1/2	1/2	0	1	1	0
2	2	1	1/4	1/4	1/2	0	0	1/2

Avalanche properties of the $P_{64;192}$ and $S_{64;192}$ boxes

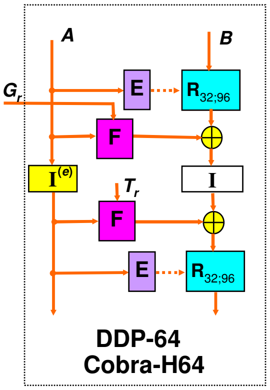


Examples of the round transformation





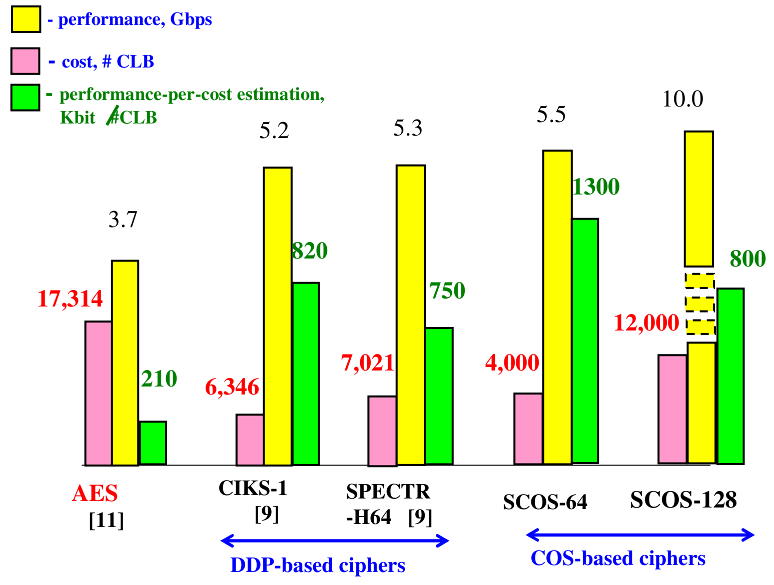
SCOS-1



DDP-64
Cobra-H64

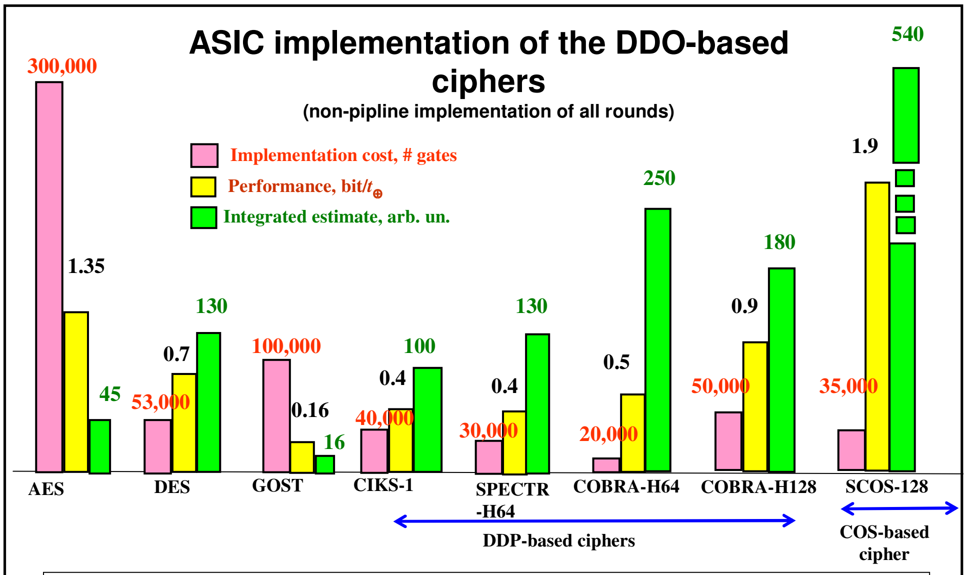
Pipeline Architecture implementation

(FPGA Xilinx Vitrex Device)



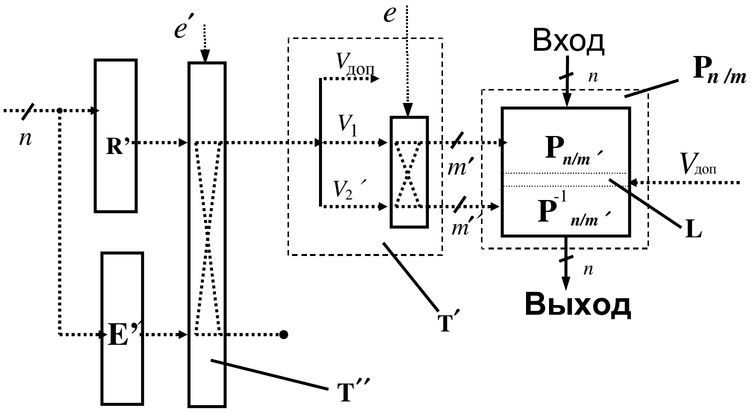
ASIC implementation of the DDO-based ciphers

(non-pipline implementation of all rounds)



Integrated estimate corresponds to the model "Performance per Cost". Integrated estimate allows copmaring ciphers with different size of input block, 64-bit ciphers (DES, GOST, CIKS-1, SPECTR-H64, Cobra-H64) and 128-bit ones (AES, Cobra-H128, SCOS-128), and different implementation cost.

Architecture of the universal bit permutation instruction



Possibility to implement in one cycle i) arbitrary given permutations, ii) data-driven permutations.

Papers on the block cipher design

- International Journal of Network security, 2006, no 2; 2006, no 3; 2007, no 1. , 2009
- Telecommunication Systems, 2006, no 2/3.
- *Mobile Networks and Applications*. 2005.
- Journal of cryptology 2002.
- Cryptologia 1998, no 1; 1998, no 2.
- Springer-Verlag Lecture Notes 2001, 2003, 2007

Books:

Innovative cryptography.- Charles River Media, Boston, Massachusetts, 2006.- 386 pp.

Data-driven ciphers for fast telecommunication systems. . – Auerbach Publications. Talor&Francis Group. New York. 2007. -185 p.

Results on digital signature algorithm design - II

1. It has been developed a way to define groups if finite vector spaces that are efficient for fast digital signature algorithm design.
2. Finding roots in the known order groups has been introduced and justified as cryptographic primitive.
3. It has been developed a way to define finite fields in finite vector spaces, in which the multiplication operation is parallelizable and has lower complexity.
4. A set of multi-signature protocols has been proposed and justified.
5. A class of provably secure DS systems has been considered, which generalizes Rabin's cryptosystem.

Generalization of Rabin's scheme

Public encryption:

$$C = M^k \bmod n$$

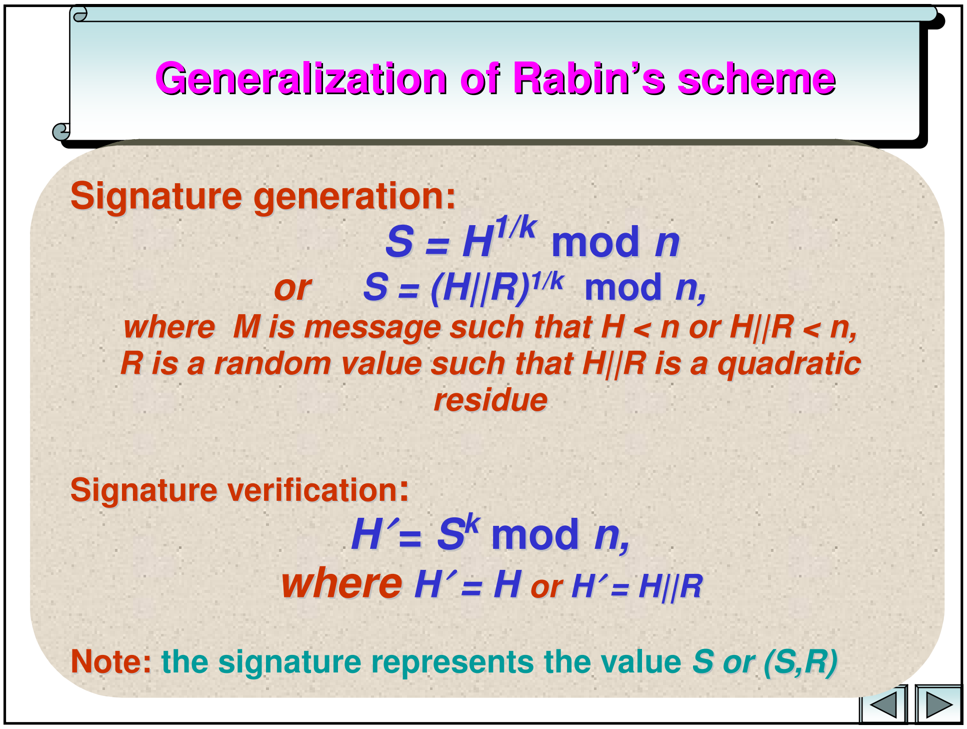
$$or\ \ C = (M||H)^k \bmod n,$$

where M is message such that M < n or M||H < n
n is the public key

Decryption:

$$M' = C^{1/k} \bmod n,$$

where the congruence has four different solutions from which one is selected as correct decrypted message, for example M′ = M||H



Generalization of Rabin's scheme

Signature generation:

$$S = H^{1/k} \bmod n$$

or $S = (H||R)^{1/k} \bmod n,$

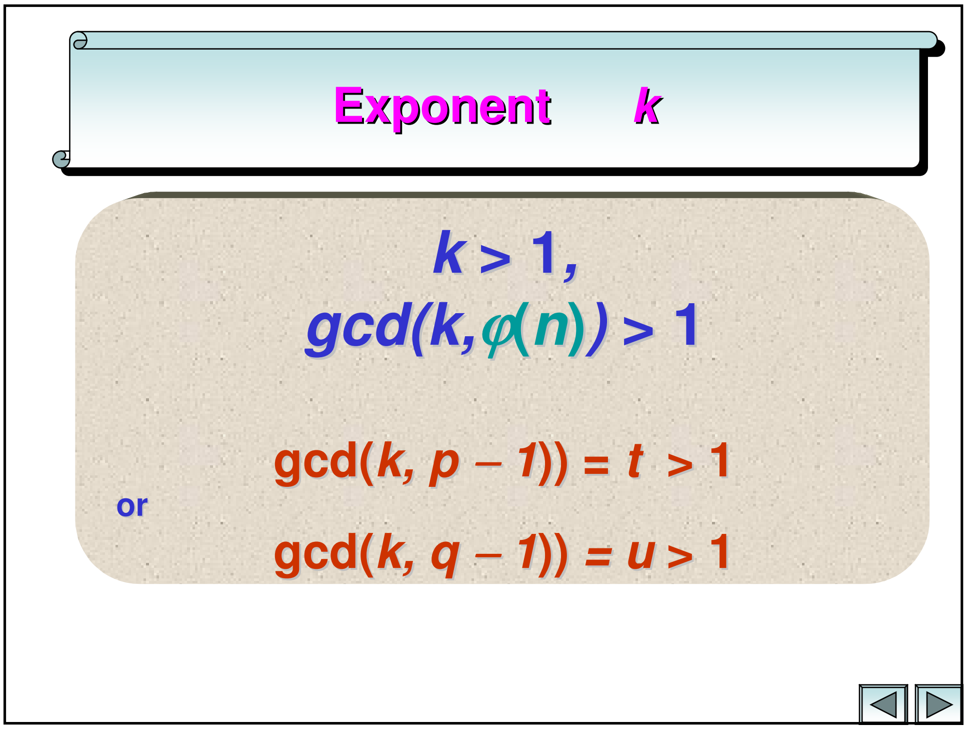
where M is message such that $H < n$ or $H||R < n$, R is a random value such that $H||R$ is a quadratic residue

Signature verification:

$$H' = S^k \bmod n,$$

where $H' = H$ or $H' = H||R$

Note: the signature represents the value *S or (S,R)*



Exponent *k*

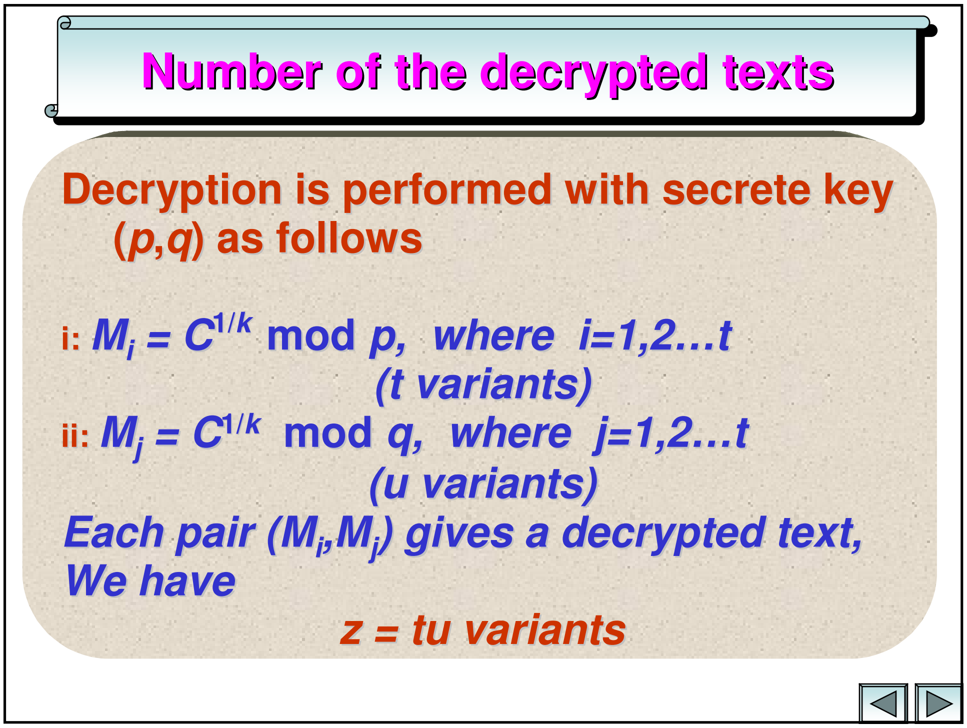
$$k > 1,$$

$$gcd(k,\varphi(n)) > 1$$

$$\gcd(k, p - 1)) = t > 1$$

or

$$\gcd(k, q - 1)) = u > 1$$



Number of the decrypted texts

Decryption is performed with secrete key (p,q) as follows

i: $M_i = C^{1/k} \bmod p$, *where* $i=1,2\ldots t$

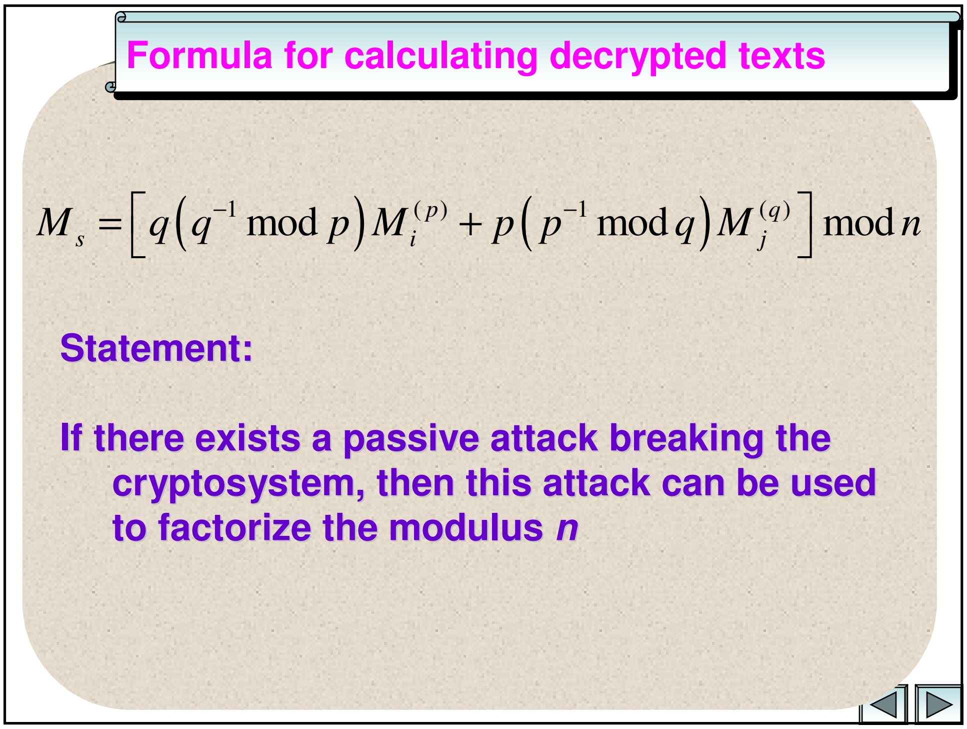
(t variants)

ii: $M_j = C^{1/k} \bmod q$, *where* $j=1,2\ldots t$

(u variants)

Each pair (M_i,M_j) gives a decrypted text, We have

$z = tu$ *variants*

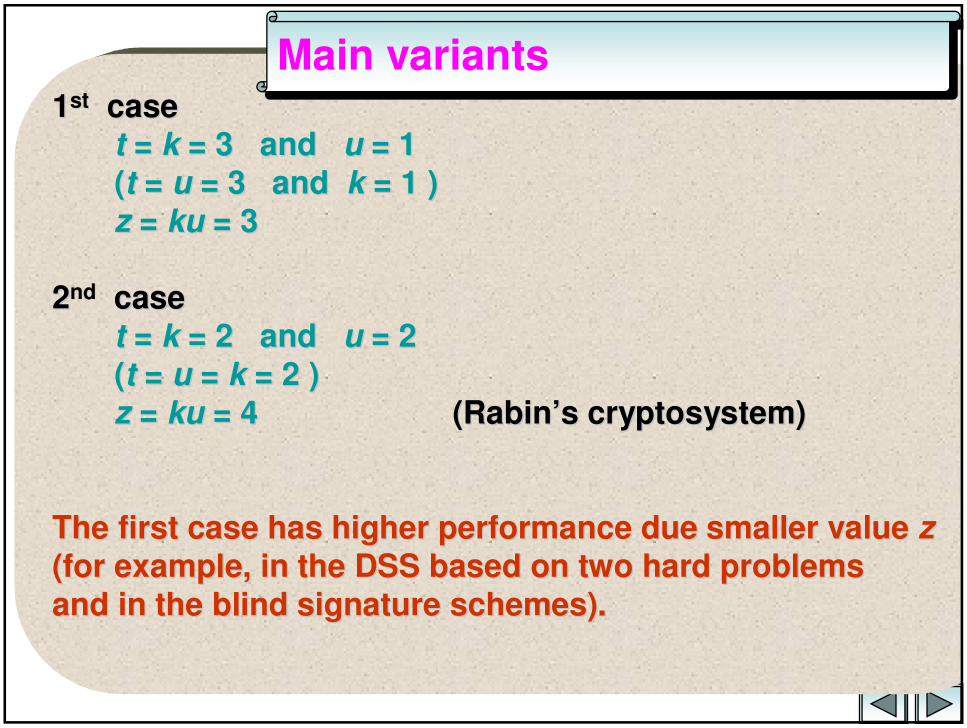


Formula for calculating decrypted texts

$$M_s = \left[q\left(q^{-1} \bmod p\right) M_i^{(p)} + p\left(p^{-1} \bmod q\right) M_j^{(q)} \right] \bmod n$$

Statement:

If there exists a passive attack breaking the cryptosystem, then this attack can be used to factorize the modulus *n*



Main variants

1st case

$t = k = 3$ and $u = 1$

($t = u = 3$ and $k = 1$)

$z = ku = 3$

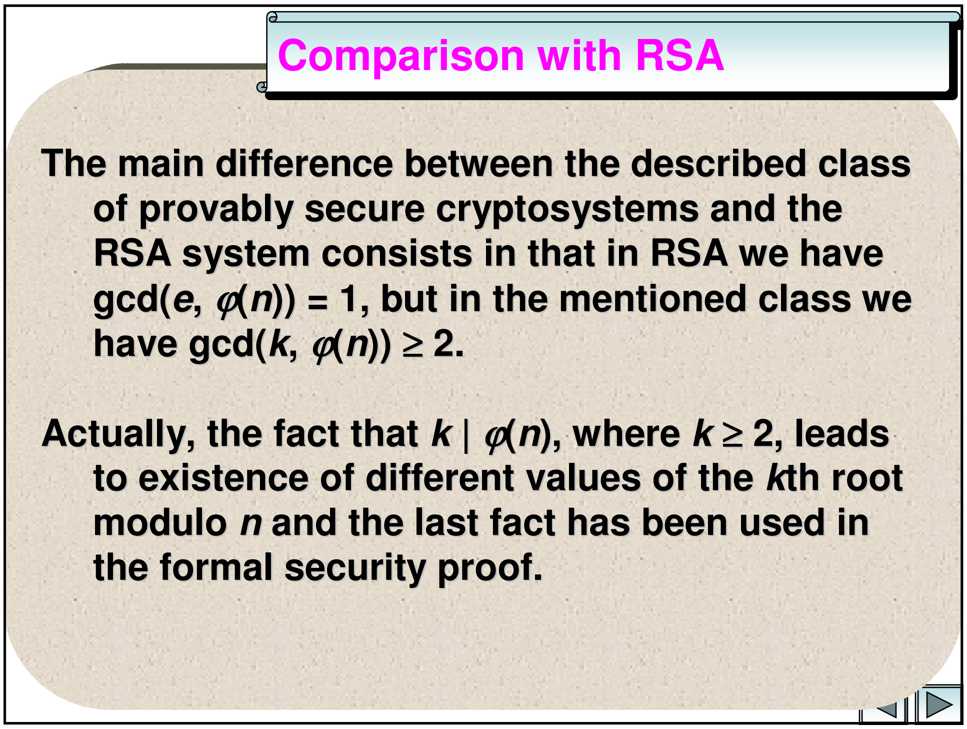
2nd case

$t = k = 2$ and $u = 2$

($t = u = k = 2$)

$z = ku = 4$ (Rabin's cryptosystem)

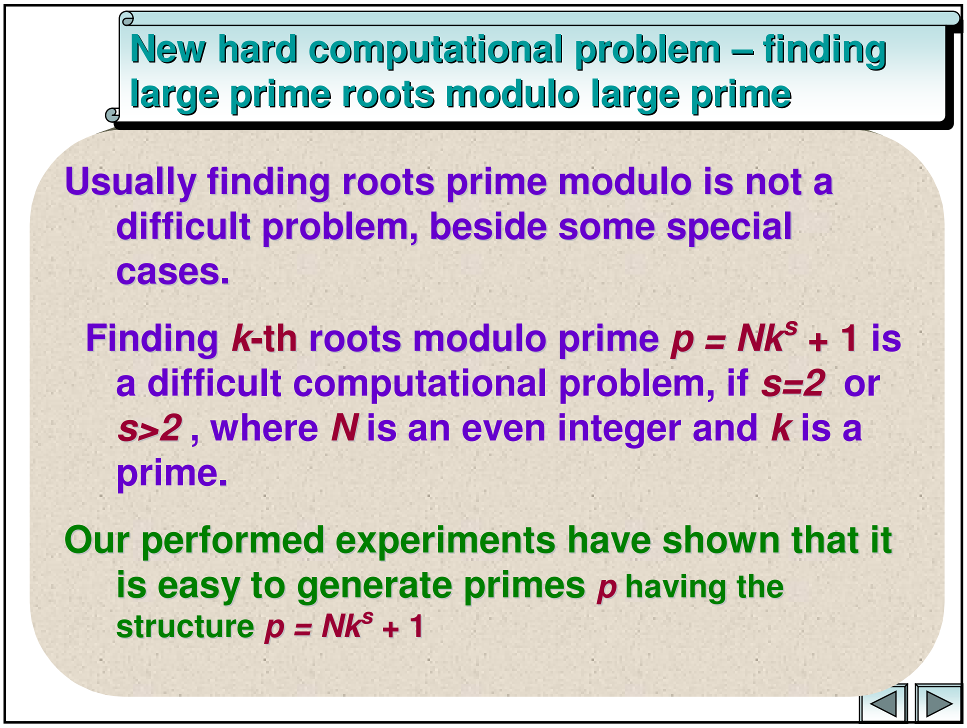
The first case has higher performance due smaller value z (for example, in the DSS based on two hard problems and in the blind signature schemes).



Comparison with RSA

The main difference between the described class of provably secure cryptosystems and the RSA system consists in that in RSA we have gcd(e, $\varphi(n)$) = 1, but in the mentioned class we have gcd(k, $\varphi(n)$) $\geq$ 2.

Actually, the fact that $k \mid \varphi(n)$, where $k \geq 2$, leads to existence of different values of the kth root modulo n and the last fact has been used in the formal security proof.

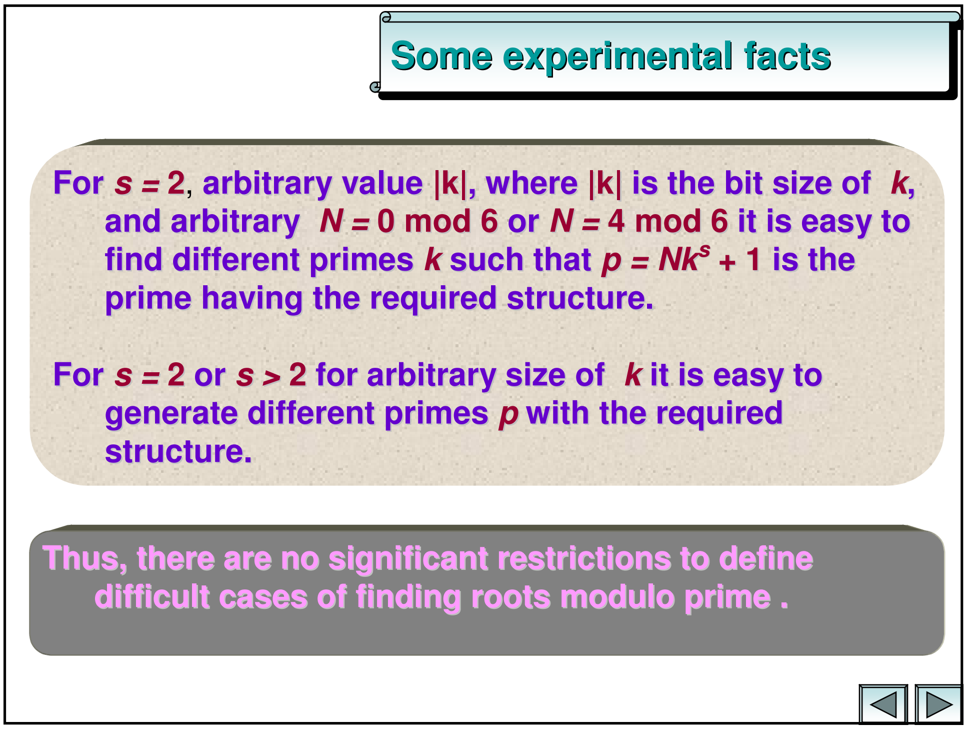


New hard computational problem – finding large prime roots modulo large prime

Usually finding roots prime modulo is not a difficult problem, beside some special cases.

Finding *k*-th roots modulo prime $p = Nk^s + 1$ is a difficult computational problem, if $s=2$ or $s>2$, where *N* is an even integer and *k* is a prime.

Our performed experiments have shown that it is easy to generate primes *p* having the structure $p = Nk^s + 1$



Some experimental facts

For $s = 2$, arbitrary value |k|, where |k| is the bit size of *k*, and arbitrary $N = 0$ mod 6 or $N = 4$ mod 6 it is easy to find different primes *k* such that $p = Nk^s + 1$ is the prime having the required structure.

For $s = 2$ or $s > 2$ for arbitrary size of *k* it is easy to generate different primes *p* with the required structure.

Thus, there are no significant restrictions to define difficult cases of finding roots modulo prime .

Peculiarities of the DS algorithms based on the proposed difficult problem

The public key is computed as follows $y=x^k \bmod p$, where x is the secrete key and y is the publik key.

The DS algorithms are randomized, i.e. while generating a signature a disposable random secrete key t is used to compute a randomized signature part $R=t^k \bmod p$

The second part of the signature S is computed depending on the secrete key x, the disposable secrete key k, and the hash function H of the message to be signed.

◁ ▷

The DS algorithm based on finding roots modulo prime

Signature generation:

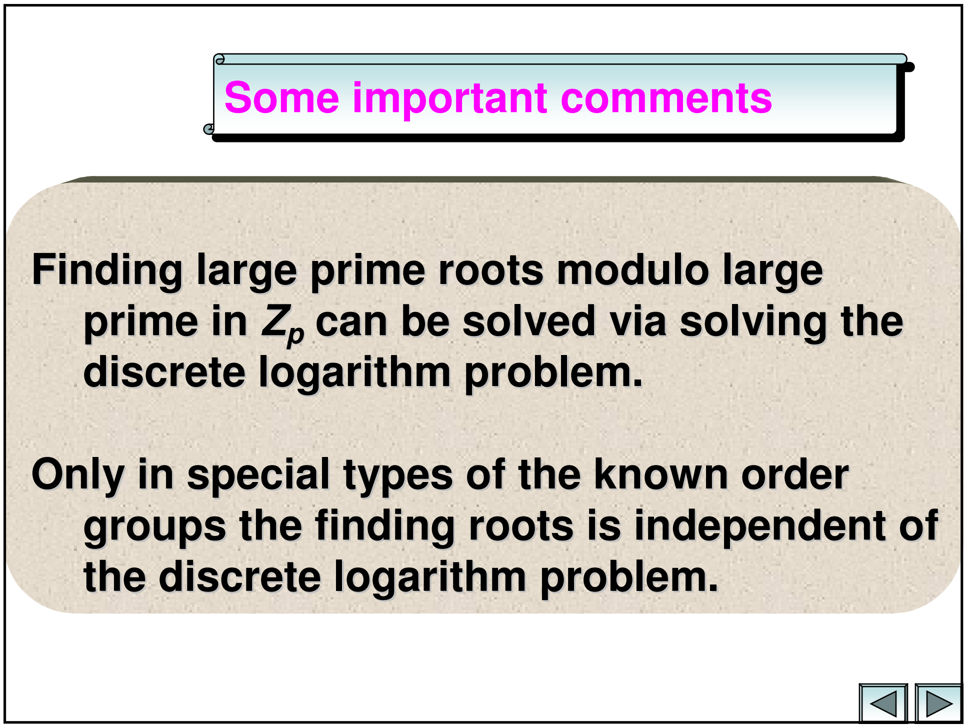
1) *generate a random value* $t < p$
2) *calculate* $R = t^k \bmod p$ *and the first part of the signature* $E=F(R,M)$, *where* M *is the document to be signed*
3) *calculate the second part of the signature* $S = x^E t \bmod p$

<u>Output:</u> *signature* (E,S)

Signature verification:

1) *compute* $R' = S^k y^{-E} \bmod p$
2) *compute* $E' = F(R', M)$
3) *compare* E' *and* E; *if* $E' = E$, *then the signature* (E,S) *is valid*

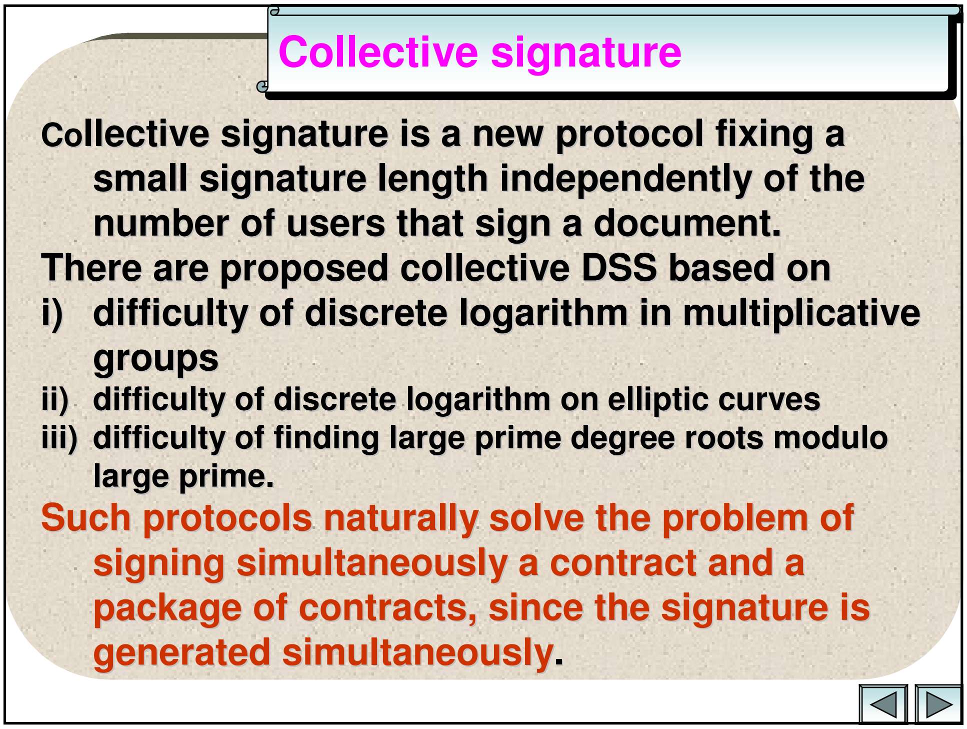
◁ ▷



Some important comments

Finding large prime roots modulo large prime in Z_p can be solved via solving the discrete logarithm problem.

Only in special types of the known order groups the finding roots is independent of the discrete logarithm problem.



Collective signature

Collective signature is a new protocol fixing a small signature length independently of the number of users that sign a document.

There are proposed collective DSS based on

i) difficulty of discrete logarithm in multiplicative groups

ii) difficulty of discrete logarithm on elliptic curves

iii) difficulty of finding large prime degree roots modulo large prime.

Such protocols naturally solve the problem of signing simultaneously a contract and a package of contracts, since the signature is generated simultaneously.

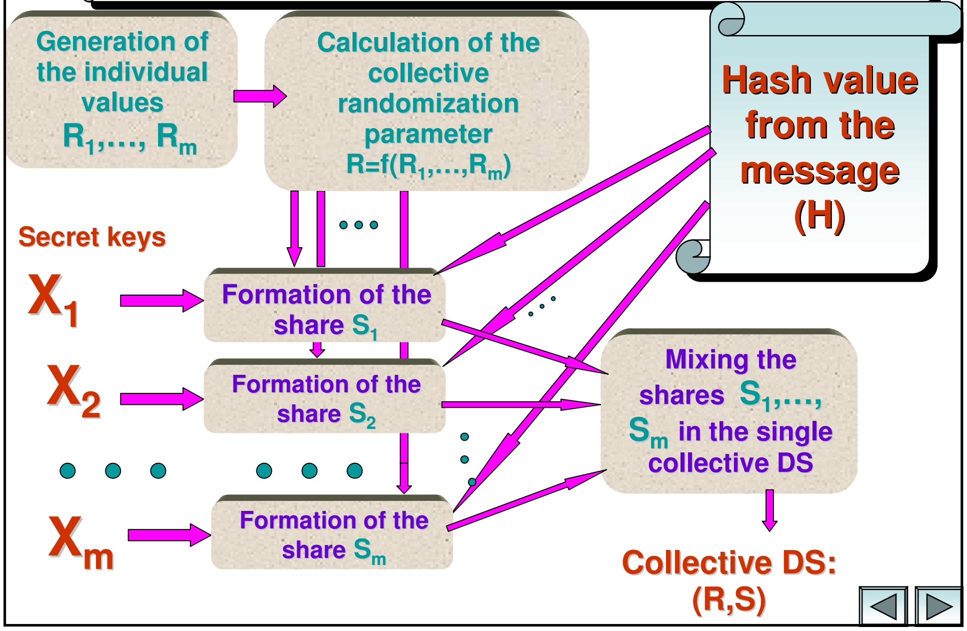
Collective digital signatures formation

In such protocols there ia used a notin of the collective public key computed from the individual public key of the signers. :

$$Y = f(Y_1, Y_2,\dots,Y_m)$$

where m is the number of signers owning the public keys Y1, Y2,…,Ym

Collective DS formation scheme



Generation of the individual values $R_1,\dots, R_m$

Calculation of the collective randomization parameter $R=f(R_1,\dots,R_m)$

Hash value from the message (H)

Secret keys

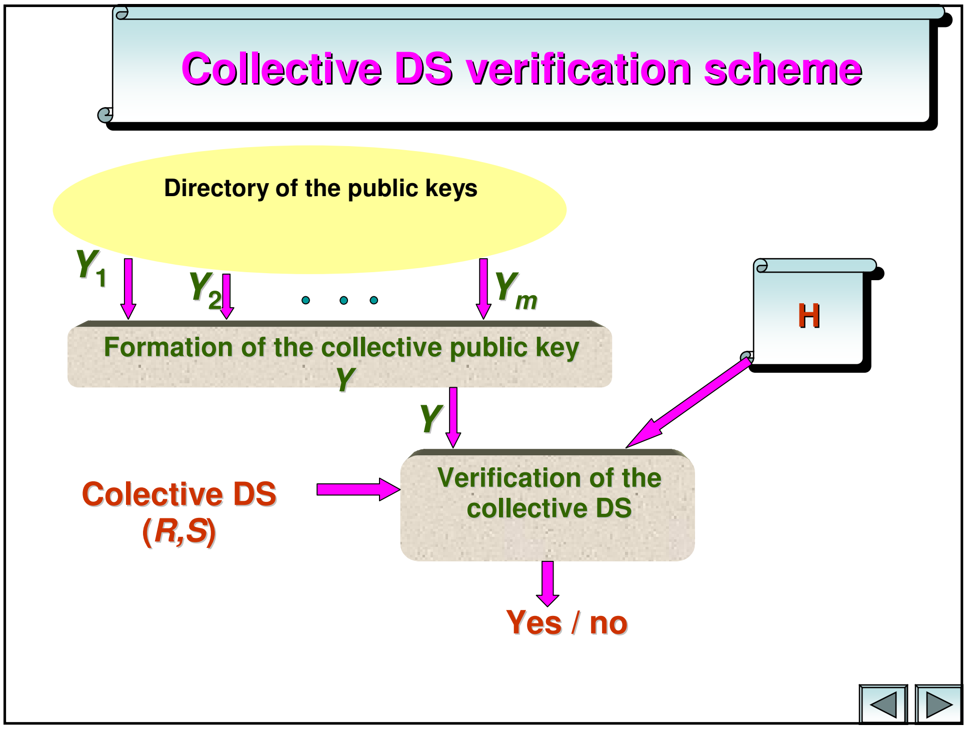
X_1 → Formation of the share S_1

X_2 → Formation of the share S_2

X_m → Formation of the share S_m

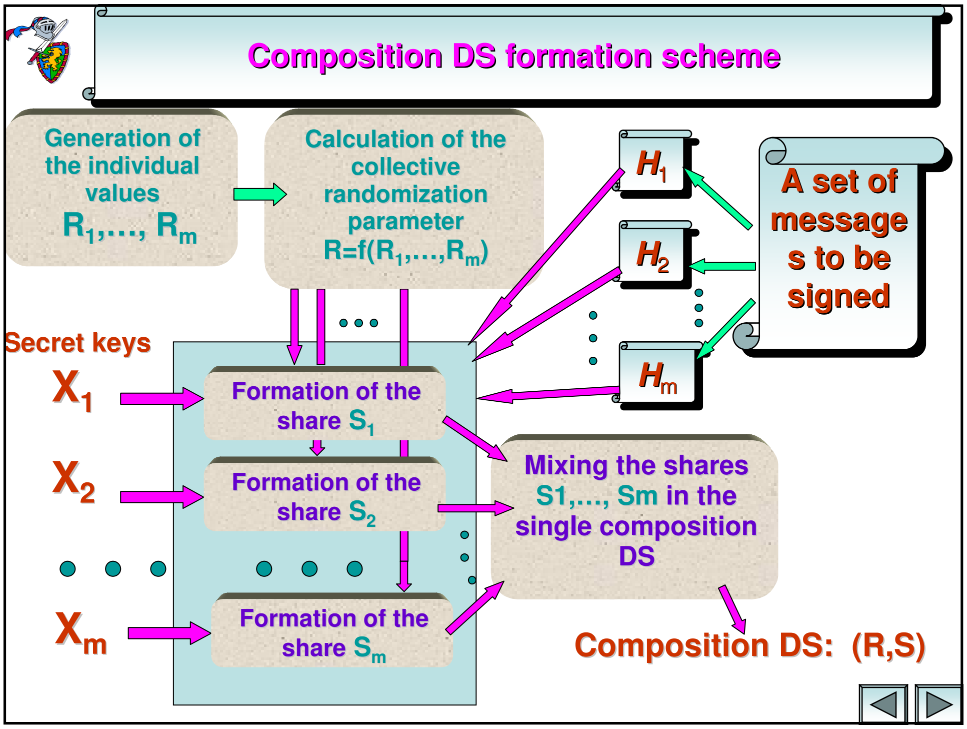
Mixing the shares $S_1,\dots,S_m$ in the single collective DS

Collective DS: (R,S)



Collective DS verification scheme

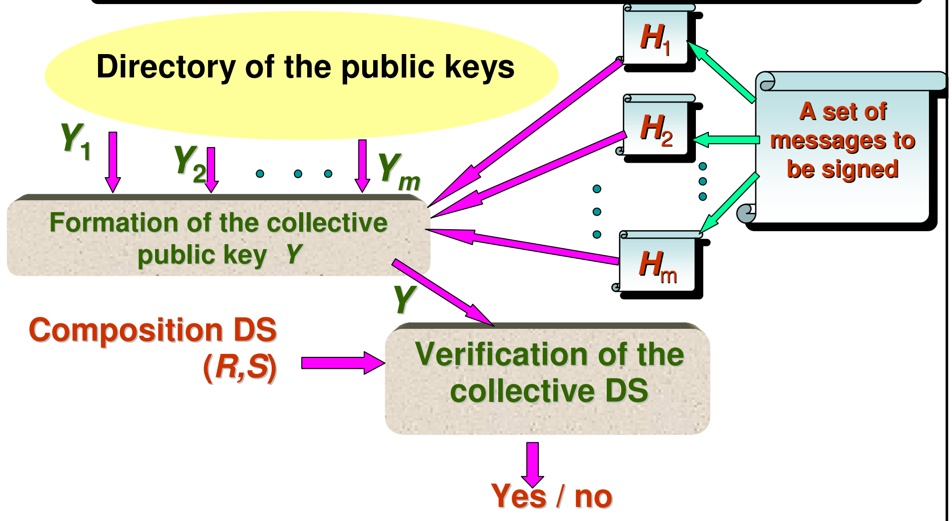
Directory of the public keys
Y_1
Y_2
. . .
Y_m
Formation of the collective public key Y
Y
H
Colective DS (R,S)
Verification of the collective DS
Yes / no



Composition DS formation scheme

Generation of the individual values R_1,…, R_m
Calculation of the collective randomization parameter R=f(R_1,…,R_m)
…
H_1
H_2
H_m
A set of messages to be signed
Secret keys
X_1
X_2
X_m
Formation of the share S_1
Formation of the share S_2
Formation of the share S_m
Mixing the shares S1,…, Sm in the single composition DS
Composition DS: (R,S)

Composition DS verification scheme


Directory of the public keys
Y_1
Y_2
. . .
Y_m
Formation of the collective public key Y
H_1
H_2
H_m
A set of messages to be signed
Y
Composition DS (R,S)
Verification of the collective DS
Yes / no

Developed types of the multi-signature protocols

- Blind collective signatures
- Collective signatures based breaking of which requires solving two computationally difficult problems

The proposed protocols provide efficient solution of the problem of signing simultaneously a package of contracts

Defining finite vector spaces for cryptographic applications

Russian patent application (december 2007)

Definition of the elements:

$$(a,b,\ldots,c) \equiv a\cdot e + b\cdot i + \ldots + c\cdot j$$

a,b,…,c are elements of a finite field

e,i,…,j are the basis vectors

Addition (+) is performed as addition of the respective coordinates

Multiplication (·) is performed using the basis-vector multiplication tables

Examples of the BVMT

In quaternion algebra there is used the following BVMT

×	e	i	j	k			
e	e	i	j	k			
i	i	– e	k	– j			
j	j	– k	– e	i			
k	k	j	– i	– e			

Commutativity and associativity properties of the BVMT imparts the same properties to the vector multiplication operation

Peculiarities of the used BVMTs

In the used BVMTs we have used the expansion coefficients

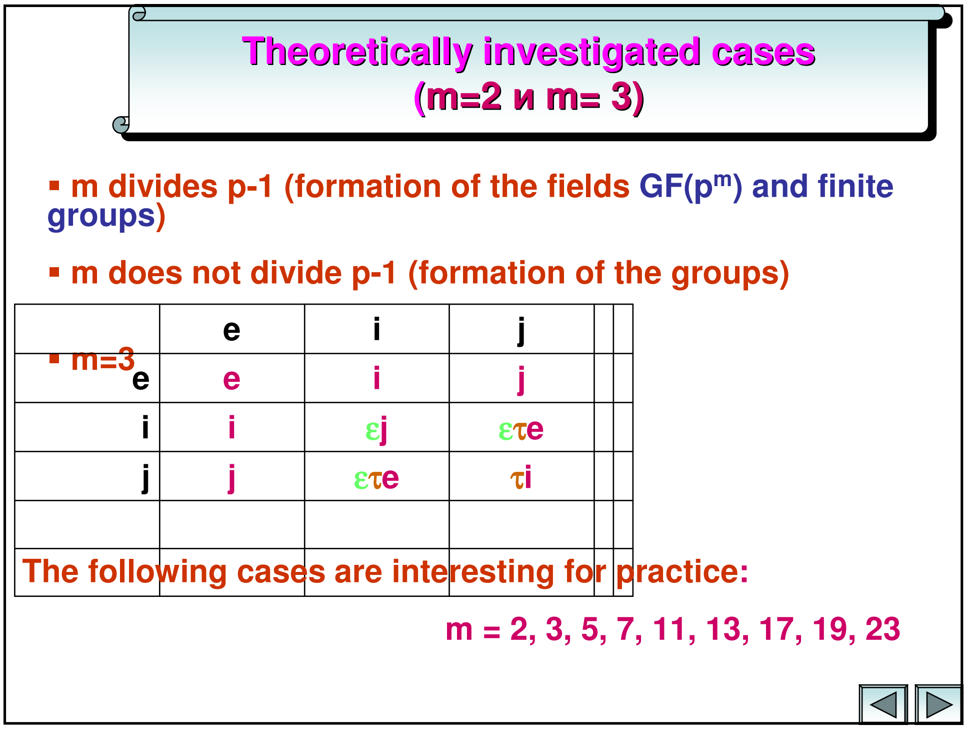
Example

×	e	i	j	k	u	v	w
e	e	i	j	k	u	v	w
i	i	εj	ε k	ε u	ε v	ε w	ε λe
j	j	ε k	ε u	ε v	ε w	ε λ e	λ i
k	k	ε u	ε v	ε w	ε λ e	λ i	λ j
u	u	ε v	ε w	ε λ e	λ i	λ j	λ k
v	v	ε w	ε λ e	λ i	λ j	λ k	λ u
w	w	ε λ e	λ i	λ j	λ k	λ u	λ v

The 7-dimension finite vector space

Two other distributions of the expansion coefficients providing commutative and associative multiplication

×	e	i	j	k	u	v	w
e	e	i	j	k	u	v	w
i	i	μj	τk	μu	τv	μw	τμe
j	j	τk	τu	τv	τw	τμe	τi
k	k	μu	τv	μw	τμe	μi	μj
u	u	τv	τw	τμe	τi	μj	τk
v	v	μw	τμe	μi	μj	μk	μu
w	w	τμe	τi	μj	τk	μu	τv



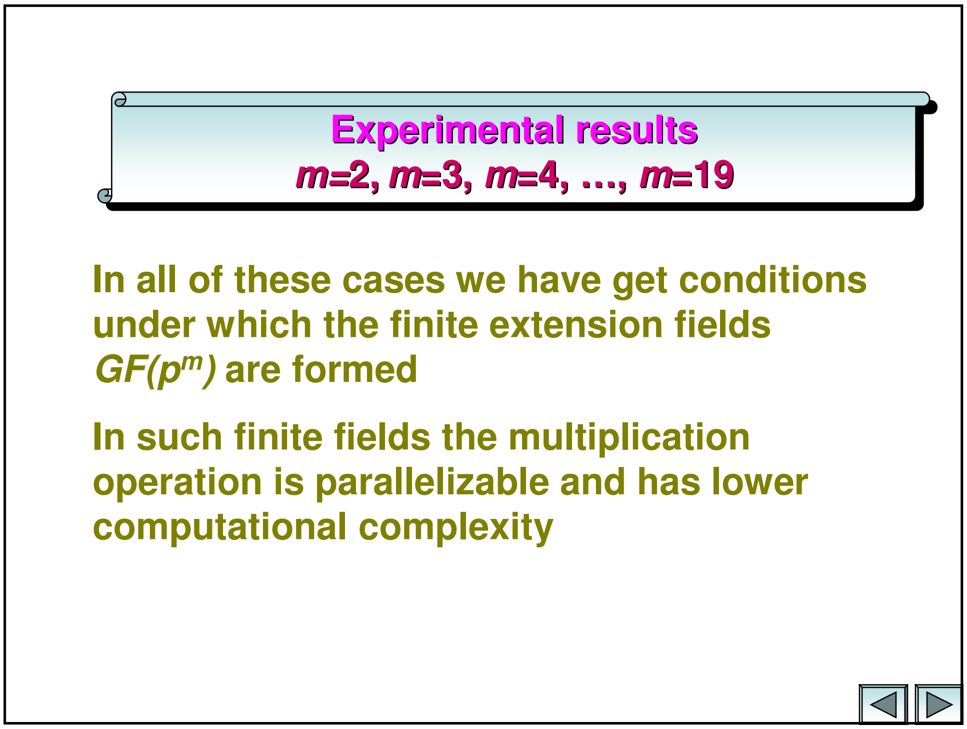
Theoretically investigated cases (m=2 и m= 3)

- m divides p-1 (formation of the fields GF(p^m) and finite groups)
- m does not divide p-1 (formation of the groups)
- m=3

	e	i	j		
e	e	i	j		
i	i	εj	ετe		
j	j	ετe	τi		

The following cases are interesting for practice:

m = 2, 3, 5, 7, 11, 13, 17, 19, 23



Experimental results *m*=2, *m*=3, *m*=4, …, *m*=19

In all of these cases we have get conditions under which the finite extension fields *GF(p^m)* are formed

In such finite fields the multiplication operation is parallelizable and has lower computational complexity

Thank you for attention