

b bits b bits Bitwise XOR Vernam Cipher (1917) $f: m \in \{0, 1\}; k \neq rand \{0, 1\}, B; k = 1.$ ⊕ = is selfinvorse С m = c - k $C = m \oplus k$ $m = C \oplus k = m \oplus k \oplus k =$ $\begin{array}{c|c} m & k & C = m \oplus k \\ \hline 0 & 0 & 0 \end{array}$ $= M \oplus 0 = M = 1$ 0 1 1 1 0 1 - - - - - K2K1 K0 Encryption of multiple bits: m; k: C: 0/01 100/ 00/1 k: Decryption - " -100110110110 M: Block cipher AES - 128, 192, 256 --> Encryption --> Decryption Advanced Encryption Standard ~ 2000 Key length 128, 192, 256, Bits: k e { 1286, 1926, 2566 } Block Cipher: Electronic Code Book -ECB mode of encryption: 1 Byte = 8 bitsK = 128 bits = 16 Bytes Data to be incripted : message m B3 ---- Bi 16 Bytes Bn B2 P_2 Key Encryption Key Encryption Key Encryption Key Decryption Key Decryption Key Decryption P1 P2 Decryption Encryption The length of any block Bi should be |Bi| = 128 bits 192 bits $|B_i| = |k| = 128 \text{ bits} = 2^7 \text{ bits}$ 256 bits EncAES(k, B1) = C1n-rillrall llr



Public Para	me	ters	PP :	q) =	, g)		Р	~ 2	204	l8 \approx	10	760	; ,	D] -	= 20	2И8Ь,	
			-	12	- 01		= 760 der diaits										
110 will		. 1	1-	1_		, <u> </u>	ta								10		
NE WIEL	W	e	Р	=	28	Бі	15,										
To genur	eat.	e I	Pr b	a	nd	Ри	\mathcal{V}	We	Ne	ed	to	ge	ene	ra	te	$PP = (p_{g}g)$	
		•				v						-					
r K = x < ra	and	==	> Pi	JK =	a =	g^ r	nod	р									
Open S	SL	. 5	oft	ware	e				Pr	k	_	20	48	Ьi	ts	F1 220487	
Pytho	0								Pul		=	20	48	bi	ts	Lig Z J	
Go								•	v	•		-		•			
							RSA	cry	ntos	vsta	m						
						Rv	est-	Shar	nir-A	ddle	emai	า					
			N 4	1.2	. f .		C: _1	L ? - I				م ما	•				
ncryption, Si	gnat	ture,	Ivias	King	for :	cont	fider fider	ntial	e-mo	oney ting	WIT	ndra	wing				
					101	com	nucl	iciai		ung							
Iultiplication ⁻	Tab.		715														
	*	1	2	3	4	5	6	7	8	9	10	11	12	13	14		
	1	1	2	3	4	5	6	7	8	9	10	11	12	13	14		
	2	2	4	6	8	10	12	14	1	3	5	7	9	11	13		
	3	3	6	9	12	0	3	6	9	12	0	3	6	9	12		
	4	4	8	12	1	5	9	13	2	6	10	14	3	7	11		
	5	5	10	0	5	10	0	5	10	0	5	10	0	5	10		
	6 7	6 7	17	3	9 12	5	6 12	12	3 11	2	10	6 2	12	3 1	9		
	, 8	7 8	14	9	2	10	3	4 11	4	12	10	13	6	14	7		
	9	9	3	12	6	0	9	3	12	6	0	9	3	12	6		
	10	10	5	0	10	5	0	10	5	0	10	5	0	10	5		
	11	11	7	3	14	10	6	2	13	9	5	1	12	8	4		
	12	12	9	6	3	0	12	9	6	3	0	12	9	6	3		
	13	13	11	9	7	5	3	1	14	12	10	8	6	4	2		
	14	14	13	12	11	10	9	8	7	6	5	4	3	2	1		
kponent Tab.		Z15															
۰ ۸	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14		
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1		
2	1	2	4	8	1	2	4	8	1	2	4	8	1	2	4		
3	1	3	9	12	6	3	9	12	6	3	9	12	6	3	9		
1	1	4	1	4	1	4	1	4	1	4	1	4	1	4	1		
4	_																
5	1	5	10	5	10	5	10	5	10	5	10	5	10	5	10		
	Public Para Ne will To gener rK = x < ra Open s Python Go ncryption, Si Aultiplication xponent Tab. A 1 2 3	Public Paramer	Public Parameters $We will use $ $To generate $ $To generate $ $rK = x < randi ==$ $Open SSL < S$ $Python $ Go ncryption, Signature, $Multiplication Tab. $ 1 1 2 3 4 4 5 6 7 8 8 9 9 10 11 12 13 13 14 14	Public Parameters PP = We will use $ p $ To generate Friend rK = x <randi ==""> Pu Open SSL soft Python Go ncryption, Signature, Mas Nultiplication Tab. Z15 * 1 2 2 2 4 3 3 6 4 4 8 5 5 10 6 6 12 7 7 14 8 8 1 9 9 3 10 10 5 11 11 7 12 12 9 13 13 11 14 14 13 xponent Tab. Z15 (xponent Tab. Z15) (xponent Tab. Z15) (xpone</randi>	Public Parameters PP = (p) We will use $ p =$ To generate Frk and rK = x < randi ==> PuK = Open SSL softward Python Generate To generate Python Generate 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 1 2 4 3 6 3 6 3 1 2 4 3 1 1 1 2 4 3 1 1 1 2 9 3 1 3 1 1 1	Public Parameters PP = (p, g) We will use $ p = 28$ To generate Prk and rK = x < randi ==> PuK = a = Open SSL software rython Go Nultiplication Tab. Z15 * 1 2 3 4 1 2 3 4 2 4 8 12 1 * 1 2 3 4 1 2 3 4 2 4 8 12 1 * 1 2 3 4 2 2 4 6 8 3 3 6 9 12 * 1 2 3 4 2 2 4 6 8 3 3 6 9 12 * 1 2 3 9 2 9 3 12 6 3 1 1 1 7 14 14 13 12 <td>Public Parameters PP = (p, g) Me will use $p = 28$ bi To genurate Prk and Pu rK = a = g* r Open SSL software PuK = a = g* r Open SSL software Python Gre Nultiplication Tab. Z15 I 2 Nultiplication Tab. Z15 I Nultiplication Tab. Z15 I Nultiplication Tab. Z15 I I 2 Nultiplication Tab. Z15 I I I I I I I I I I I I I I I I I I I <</td> <td>Public Parameters PP = (p, g) P Me will use $p = 28$ bits. To generate Prk and Puk To generate Software Python Go Rvest- Open SSL software Python Go SA Autiplication Tab. Z15 I I 1 2 3 4 5 6 1 1 2 3 4 5 6 1 1 2 3 4 5 6 1 1 2 3 4 5 6 2 2 4 6 10 12 3 6 9 12 0 3 4 8 12 1 5 9 5 5 10 0 5 10 0 6 12 3 9 3 12 0 3 9 3 12 6 3 0 12 8 1 9 1 3 0 12 10 10 5<</td> <td>Public Parameters PP = (p, g) $p \sim 2$ Me will use $[p] = 28$ bits. To generate Prk and Puk we To generate Prk and Puk we rK = x < randi ===> PuK = $a = g^x \mod p$ Open SSL software [Python [Go RSA cry Rvest-Shar ncryption, Signature, Masking: for confidential for confidential nultiplication Tab. Z15 * 1 2 3 4 5 6 7 1 1 2 3 4 5 6 7 1 2 3 4 5 6 7 1 2 3 4 5 6 7 1 2 3 4 5 6 7 1 2 3 4 5 6 7 2 2 4 6 10 12 14 3 6 9 12 0 3 6 4 8 1 9 2 10 3 11</td> <td>Public Parameters PP = (p, g) $P \sim 2^{20^4}$ We will use $[p] = 28$ bits. To genurate Frk and Puk we need $rK = x <-r randi ==> PuK = a = g^x \mod p$ Open SSL software Puk Python Pul Go RSA cryptos Rvest-Shamir-A ncryption, Signature, Masking: for confidential e-mo for confidential</td> <td>Public Parameters PP = (p, g) $P \sim 2^{2048} \approx$ We will use $p = 28$ bits. To genurate Prk and Puk we need rK = x < randi ==> PuK = a = g^x mod p Open SSL software (Prk) rython Puk Ge RSA cryptosyste Rvest-Shamir-Addle ncryption, Signature, Masking: for confidential e-money for confiden</td> <td>Public Parameters PP = (p, g) $P \sim 2^{2048} \approx 10^{4}$ We will use $p = 28$ bits. To generate Prk and Puk we need to rK = x < randi ==> PuK = a = g^x mod p Open SSL software $Prk =$ Python $Puk =$ Go RSA cryptosystem Rvest-Shamir-Addleman ncryption, Signature, Masking: for confidential e-money with for confidential e-voting tultiplication Tab. 215 * 1 2 3 4 5 6 7 8 9 10 1 1 2 3 4 5 6 7 8 9 10 2 2 4 6 8 10 12 14 1 3 5 3 3 6 9 12 0 3 6 9 12 0 4 4 8 12 1 5 9 13 2 6 10 5 5 10 0 5 10 0 5 10 0 5 6 6 12 3 9 0 6 12 3 9 0 7 7 14 6 13 5 12 4 11 3 10 8 8 1 9 2 10 3 11 4 12 5 9 9 3 12 6 0 9 3 12 6 0 10 10 5 0 10 5 0 10 5 0 10 11 11 7 3 14 10 6 2 13 9 5 12 12 9 6 3 0 12 9 6 3 0 13 13 11 9 7 5 3 1 14 12 10 14 14 13 12 11 10 9 8 7 6 5 sponent Tab. 215 a 0 1 2 3 4 5 6 7 8 9 10 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 3 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8</td> <td>Public Parameters PP = (p, g) $P \sim 2^{2048} \approx 10^{400}$ We will use $p = 28$ bits. To generate Prk and Puk we need to generate Prk and Puk we need to generate Prk and Puk we need to generate Prk and $Puk = 20$ Popen SSL software $Puk = 20$ RSA cryptosystem Rvest-Shamir-Addleman ncryption, Signature, Masking: for confidential e-money withdra for confidential e-voting Nultiplication Tab. 215 * 1 2 3 4 5 6 7 8 9 10 11 1 1 2 3 4 5 6 7 8 9 10 11 2 2 4 6 8 10 12 14 1 3 5 7 3 3 6 9 12 0 3 6 9 12 0 3 4 4 8 12 1 5 9 13 2 6 10 14 5 5 10 0 5 10 0 5 10 0 5 10 6 6 12 3 9 0 6 12 3 9 0 6 7 7 14 6 13 5 12 4 11 3 10 2 8 8 1 9 2 10 3 11 4 12 5 13 9 9 3 12 6 0 9 3 12 6 0 9 10 10 5 0 10 5 0 10 5 0 10 5 11 11 7 3 14 10 6 2 13 9 5 1 12 12 9 6 3 0 12 9 6 3 0 12 13 13 11 9 7 5 3 1 14 12 10 8 14 14 13 12 11 10 9 8 7 6 5 4 xponent Tab. 215 A 0 1 2 3 4 5 6 7 8 9 10 11 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 4 8 1 2 4 8 1 2 4 8 3 1 3 9 12 6 3 9 12 6 3 9 12 6 3 9 10 11 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1</td> <td>Public Parameters PP = (p, g) $P \sim 2^{2048} \approx 10^{40}$; 1 We will use $p = 28$ bits. To genurate Prk and Puk we need to gene rK = x < randi ==> PuK = a = g^x mod p Open SSL software $Prk = 2048$ Rython $Puk = 2048$ rython $Puk = 2048$ Rest-Shamir-Addleman ncryption, Signature, Masking: for confidential e-money withdrawing for confidential e-woting nultiplication Tab. Z15 * 1 2 3 4 5 6 7 8 9 10 11 12 2 2 4 6 8 10 12 14 1 3 5 7 9 3 3 6 9 12 0 3 6 9 12 0 3 6 4 4 8 12 1 5 9 13 2 6 10 14 3 5 5 10 0 5 10 0 5 10 0 5 10 0 6 6 12 3 9 0 6 12 3 9 0 6 12 7 7 14 6 13 5 12 4 11 3 10 2 9 8 8 1 9 2 10 3 11 4 12 5 13 6 9 9 3 12 6 0 9 3 12 6 0 9 3 10 10 5 0 10 5 0 10 5 0 10 5 0 11 11 7 3 14 10 6 2 13 9 5 1 12 12 12 9 6 3 0 12 9 6 3 0 12 9 13 13 11 9 7 5 3 1 14 12 10 8 6 14 14 13 12 11 10 9 8 7 6 5 4 3 xponent Tab. Z15 xponent Tab. Z15 xponent</td> <td>Public Parameters PP = (p, g) $P \sim 2^{2048} \approx 10^{740}$; p . We will use p = 28 bits. To genurate Prk and Puk we need to general rk = x < randi ==> PuK = a = g* mod p Open SSL software Prk = 2048 bit Python Puk = 2048 bit Rython Puk = 2048 bit Genurate Prk and Puk lead leader RSA cryptosystem Rvest-Shamir-Addleman ncryption, Signature, Masking: for confidential e-money withdrawing for confidential e-voting tultiplication Tab. Z15 * 1 2 3 4 5 6 7 8 9 10 11 12 13 2 2 4 6 8 10 12 14 1 3 5 7 9 11 3 3 6 9 12 0 3 6 9 12 0 3 6 9 4 4 8 12 1 5 9 13 2 6 10 14 3 7 5 5 10 0 5 10 0 5 10 0 5 10 0 5 6 6 12 3 9 0 6 12 3 9 0 6 12 3 7 7 14 6 13 5 12 4 11 3 10 2 9 1 8 8 1 9 2 10 3 11 4 12 5 13 6 14 9 9 3 12 6 0 9 3 12 6 0 9 3 12 10 10 5 0 10 5 0 10 5 0 10 5 0 10 11 11 7 3 14 10 6 2 13 9 5 1 12 8 12 12 9 6 3 0 12 9 6 3 0 12 9 6 3 1 3 11 9 7 5 3 1 14 12 10 8 6 4 14 14 13 12 11 10 9 8 7 6 5 4 3 2 xponent Tab. Z15 xponent Ta</td> <td>Public Parameters PP = (p, g) $P \sim 2^{2048} \approx 10^{760}$; $p = 24$ = 76 We will use $p = 28$ bits. To generate Prk and Puk we need to generate rk = x < randi ==> PuK = a = g^x mod p Open SSL software $Prk = 2048$ bits Puthon $Puk = 2048$ bits Ge RSA cryptosystem Rvest-Shamir-Addleman ncryption, Signature, Masking: for confidential e-money withdrawing for confidential e-voting Nultiplication Tab. 215 * 1 2 3 4 5 6 7 8 9 10 11 12 13 14 1 1 2 3 4 5 6 7 8 9 10 11 12 13 14 2 4 6 8 10 12 14 1 3 5 7 9 11 13 3 3 6 9 12 0 3 6 9 12 0 3 6 9 12 4 4 8 12 1 5 9 13 2 6 10 14 3 7 11 5 5 10 0 5 10 0 5 10 0 5 10 0 5 10 0 5 10 6 6 12 3 9 0 6 12 3 9 0 6 12 3 9 7 7 14 6 13 5 12 4 11 3 10 2 9 1 8 8 8 1 9 2 10 3 11 4 12 5 13 6 14 7 9 9 3 12 6 0 9 3 12 6 0 9 3 12 6 0 10 4 5 10 1 11 7 3 14 10 6 2 13 9 5 1 12 8 4 1 2 12 9 6 3 0 12 9 6 3 0 12 9 6 3 1 3 13 11 9 7 5 3 1 14 12 10 8 6 4 2 1 4 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 1 2 1 2 9 6 3 0 12 9 6 3 0 12 9 6 3 1 3 13 11 9 7 5 3 1 14 12 10 8 6 4 2 1 4 1 1 1 2 11 10 9 8 7 6 5 4 3 2 1 xponent Tab. 2 12 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 3 1 3 9 12 6 3 9 12 6 3 9 12 6 3 9 12 6 3 9 12 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1</td>	Public Parameters PP = (p, g) Me will use $ p = 28$ bi To genurate Prk and Pu rK = a = g* r Open SSL software PuK = a = g* r Open SSL software Python Gre Nultiplication Tab. Z15 I 2 Nultiplication Tab. Z15 I Nultiplication Tab. Z15 I Nultiplication Tab. Z15 I I 2 Nultiplication Tab. Z15 I I I I I I I I I I I I I I I I I I I <	Public Parameters PP = (p, g) P Me will use $ p = 28$ bits. To generate Prk and Puk To generate Software Python Go Rvest- Open SSL software Python Go SA Autiplication Tab. Z15 I I 1 2 3 4 5 6 1 1 2 3 4 5 6 1 1 2 3 4 5 6 1 1 2 3 4 5 6 2 2 4 6 10 12 3 6 9 12 0 3 4 8 12 1 5 9 5 5 10 0 5 10 0 6 12 3 9 3 12 0 3 9 3 12 6 3 0 12 8 1 9 1 3 0 12 10 10 5<	Public Parameters PP = (p, g) $p \sim 2$ Me will use $[p] = 28$ bits. To generate Prk and Puk we To generate Prk and Puk we rK = x < randi ===> PuK = $a = g^x \mod p$ Open SSL software [Python [Go RSA cry Rvest-Shar ncryption, Signature, Masking: for confidential for confidential nultiplication Tab. Z15 * 1 2 3 4 5 6 7 1 1 2 3 4 5 6 7 1 2 3 4 5 6 7 1 2 3 4 5 6 7 1 2 3 4 5 6 7 1 2 3 4 5 6 7 2 2 4 6 10 12 14 3 6 9 12 0 3 6 4 8 1 9 2 10 3 11	Public Parameters PP = (p, g) $P \sim 2^{20^4}$ We will use $[p] = 28$ bits. To genurate Frk and Puk we need $rK = x <-r randi ==> PuK = a = g^x \mod p$ Open SSL software Puk Python Pul Go RSA cryptos Rvest-Shamir-A ncryption, Signature, Masking: for confidential e-mo for confidential	Public Parameters PP = (p, g) $P \sim 2^{2048} \approx$ We will use $ p = 28$ bits. To genurate Prk and Puk we need rK = x < randi ==> PuK = a = g ^x mod p Open SSL software (Prk) rython Puk Ge RSA cryptosyste Rvest-Shamir-Addle ncryption, Signature, Masking: for confidential e-money for confiden	Public Parameters PP = (p, g) $P \sim 2^{2048} \approx 10^{4}$ We will use $ p = 28$ bits. To generate Prk and Puk we need to rK = x < randi ==> PuK = a = g ^x mod p Open SSL software $ Prk =$ Python $ Puk =$ Go RSA cryptosystem Rvest-Shamir-Addleman ncryption, Signature, Masking: for confidential e-money with for confidential e-voting tultiplication Tab. 215 * 1 2 3 4 5 6 7 8 9 10 1 1 2 3 4 5 6 7 8 9 10 2 2 4 6 8 10 12 14 1 3 5 3 3 6 9 12 0 3 6 9 12 0 4 4 8 12 1 5 9 13 2 6 10 5 5 10 0 5 10 0 5 10 0 5 6 6 12 3 9 0 6 12 3 9 0 7 7 14 6 13 5 12 4 11 3 10 8 8 1 9 2 10 3 11 4 12 5 9 9 3 12 6 0 9 3 12 6 0 10 10 5 0 10 5 0 10 5 0 10 11 11 7 3 14 10 6 2 13 9 5 12 12 9 6 3 0 12 9 6 3 0 13 13 11 9 7 5 3 1 14 12 10 14 14 13 12 11 10 9 8 7 6 5 sponent Tab. 215 a 0 1 2 3 4 5 6 7 8 9 10 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 3 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8	Public Parameters PP = (p, g) $P \sim 2^{2048} \approx 10^{400}$ We will use $ p = 28$ bits. To generate Prk and Puk we need to generate Prk and Puk we need to generate Prk and Puk we need to generate Prk and $Puk = 20$ Popen SSL software $ Puk = 20$ RSA cryptosystem Rvest-Shamir-Addleman ncryption, Signature, Masking: for confidential e-money withdra for confidential e-voting Nultiplication Tab. 215 * 1 2 3 4 5 6 7 8 9 10 11 1 1 2 3 4 5 6 7 8 9 10 11 2 2 4 6 8 10 12 14 1 3 5 7 3 3 6 9 12 0 3 6 9 12 0 3 4 4 8 12 1 5 9 13 2 6 10 14 5 5 10 0 5 10 0 5 10 0 5 10 6 6 12 3 9 0 6 12 3 9 0 6 7 7 14 6 13 5 12 4 11 3 10 2 8 8 1 9 2 10 3 11 4 12 5 13 9 9 3 12 6 0 9 3 12 6 0 9 10 10 5 0 10 5 0 10 5 0 10 5 11 11 7 3 14 10 6 2 13 9 5 1 12 12 9 6 3 0 12 9 6 3 0 12 13 13 11 9 7 5 3 1 14 12 10 8 14 14 13 12 11 10 9 8 7 6 5 4 xponent Tab. 215 A 0 1 2 3 4 5 6 7 8 9 10 11 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 4 8 1 2 4 8 1 2 4 8 3 1 3 9 12 6 3 9 12 6 3 9 12 6 3 9 10 11 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	Public Parameters PP = (p, g) $P \sim 2^{2048} \approx 10^{40}$; 1 We will use $ p = 28$ bits. To genurate Prk and Puk we need to gene rK = x < randi ==> PuK = a = g ^x mod p Open SSL software $ Prk = 2048$ Rython $ Puk = 2048$ rython $ Puk = 2048$ Rest-Shamir-Addleman ncryption, Signature, Masking: for confidential e-money withdrawing for confidential e-woting nultiplication Tab. Z15 * 1 2 3 4 5 6 7 8 9 10 11 12 2 2 4 6 8 10 12 14 1 3 5 7 9 3 3 6 9 12 0 3 6 9 12 0 3 6 4 4 8 12 1 5 9 13 2 6 10 14 3 5 5 10 0 5 10 0 5 10 0 5 10 0 6 6 12 3 9 0 6 12 3 9 0 6 12 7 7 14 6 13 5 12 4 11 3 10 2 9 8 8 1 9 2 10 3 11 4 12 5 13 6 9 9 3 12 6 0 9 3 12 6 0 9 3 10 10 5 0 10 5 0 10 5 0 10 5 0 11 11 7 3 14 10 6 2 13 9 5 1 12 12 12 9 6 3 0 12 9 6 3 0 12 9 13 13 11 9 7 5 3 1 14 12 10 8 6 14 14 13 12 11 10 9 8 7 6 5 4 3 xponent Tab. Z15 xponent	Public Parameters PP = (p, g) $P \sim 2^{2048} \approx 10^{740}$; p . We will use p = 28 bits. To genurate Prk and Puk we need to general rk = x < randi ==> PuK = a = g* mod p Open SSL software Prk = 2048 bit Python Puk = 2048 bit Rython Puk = 2048 bit Genurate Prk and Puk lead leader RSA cryptosystem Rvest-Shamir-Addleman ncryption, Signature, Masking: for confidential e-money withdrawing for confidential e-voting tultiplication Tab. Z15 * 1 2 3 4 5 6 7 8 9 10 11 12 13 2 2 4 6 8 10 12 14 1 3 5 7 9 11 3 3 6 9 12 0 3 6 9 12 0 3 6 9 4 4 8 12 1 5 9 13 2 6 10 14 3 7 5 5 10 0 5 10 0 5 10 0 5 10 0 5 6 6 12 3 9 0 6 12 3 9 0 6 12 3 7 7 14 6 13 5 12 4 11 3 10 2 9 1 8 8 1 9 2 10 3 11 4 12 5 13 6 14 9 9 3 12 6 0 9 3 12 6 0 9 3 12 10 10 5 0 10 5 0 10 5 0 10 5 0 10 11 11 7 3 14 10 6 2 13 9 5 1 12 8 12 12 9 6 3 0 12 9 6 3 0 12 9 6 3 1 3 11 9 7 5 3 1 14 12 10 8 6 4 14 14 13 12 11 10 9 8 7 6 5 4 3 2 xponent Tab. Z15 xponent Ta	Public Parameters PP = (p, g) $P \sim 2^{2048} \approx 10^{760}$; $ p = 24$ = 76 We will use $ p = 28$ bits. To generate Prk and Puk we need to generate rk = x < randi ==> PuK = a = g ^x mod p Open SSL software $ Prk = 2048$ bits Puthon $ Puk = 2048$ bits Ge RSA cryptosystem Rvest-Shamir-Addleman ncryption, Signature, Masking: for confidential e-money withdrawing for confidential e-voting Nultiplication Tab. 215 * 1 2 3 4 5 6 7 8 9 10 11 12 13 14 1 1 2 3 4 5 6 7 8 9 10 11 12 13 14 2 4 6 8 10 12 14 1 3 5 7 9 11 13 3 3 6 9 12 0 3 6 9 12 0 3 6 9 12 4 4 8 12 1 5 9 13 2 6 10 14 3 7 11 5 5 10 0 5 10 0 5 10 0 5 10 0 5 10 0 5 10 6 6 12 3 9 0 6 12 3 9 0 6 12 3 9 7 7 14 6 13 5 12 4 11 3 10 2 9 1 8 8 8 1 9 2 10 3 11 4 12 5 13 6 14 7 9 9 3 12 6 0 9 3 12 6 0 9 3 12 6 0 10 4 5 10 1 11 7 3 14 10 6 2 13 9 5 1 12 8 4 1 2 12 9 6 3 0 12 9 6 3 0 12 9 6 3 1 3 13 11 9 7 5 3 1 14 12 10 8 6 4 2 1 4 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 1 2 1 2 9 6 3 0 12 9 6 3 0 12 9 6 3 1 3 13 11 9 7 5 3 1 14 12 10 8 6 4 2 1 4 1 1 1 2 11 10 9 8 7 6 5 4 3 2 1 xponent Tab. 2 12 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 8 1 2 4 3 1 3 9 12 6 3 9 12 6 3 9 12 6 3 9 12 6 3 9 12 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	

7	1	7	4	13	1	7	4	13	1	7	4	13	1	7	4
8	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4
9	1	9	6	9	6	9	6	9	6	9	6	9	6	9	6
10	1	10	10	10	10	10	10	10	10	10	10	10	10	10	10
11	1	11	1	11	1	11	1	11	1	11	1	11	1	11	1
12	1	12	9	3	6	12	9	3	6	12	9	3	6	12	9
13	1	13	4	7	1	13	4	7	1	13	4	7	1	13	4
14	1	14	1	14	1	14	1	14	1	14	1	14	1	14	1

1. Parameters generation: p = prime; q = prime: generated at random n=p·q - RSA module $Z_n = \{0, 1, 2, ..., n-1\}; Let z \in Z_n$ If $gcd(z,n) = 1 \implies \exists! z^{-1}$ such that $z \cdot z^{-1} \mod n = 1$ E.g. Let n = 15; $z = 2 \implies z^{-1} \mod 15 = 8$ since 2.8 mod 15=1 The number of numbers having inverse values in In is defined by Euler Totient function $\phi(n) = \phi = fy$. If $n = p \cdot q$, when $p_{q} q$ -primes $\Rightarrow \phi(n) = \phi = (p-1) \cdot (q-1)$ 2. Generate number e-exponent of RSA, such that $gcd(e, \phi) = 1 \Rightarrow e$ has its multiplicative inverse element $\bar{e}^1 \mod \phi$. According to RSA standard e = 2 + 1. 3. Generate $PrK = d \Rightarrow d = \bar{e}^{\dagger} mad \phi$.

Security considerations: if p, q-are large primes => n=p.q-is large. To find Prk=d it is necessary to factor n. When n-is large it is infeasible with classical computers, \gg factor(n) $\rightarrow p_1 q \rightarrow \phi = (p-1) \cdot (q-1) \rightarrow \phi$ \gg mulinv $(e, \phi) = d$. Security relies on the complexity of factorization problem. According to Euler theorem Euler theorem. If gcd(*z*,*n*)=1 then

exponents are computing

z[•]= 1 mod **n**

RSA: PuK = (n, e); PrK = d.

Asymmetric Encryption - DecryptionAsymmetric Signing - Verification $c=Enc(PuK_A, m)$ $S=Sign(PrK_A, m)$ $m=Dec(PrK_A, c)$ $V=Ver(PuK_A, m, s), V \in \{True, False\} \equiv \{1, 0\}$ Bob : senderAlice : sender

mod P.





Search in Database is performed in the fields which are ordered. Id Îd 1 3 2 ordering 3 4 7 Dleg2(n) 32-steps 1 Ч 7 8 n-records $N \sim 2^{32}$ MDPI Symmetry 2.6 ...

Database encryption has received increased attention recently due to the enormous amount of sensitive data stored in outsourcing cloud databases. One of promising solutions to protect the confidentiality of sensitive data is to use encryption and **performing guery evaluation over**

2020

encrypted data.

Order-Revealing Encryption - OREnc

Order-Preserving Encryption. Property-preserving encryption which preserves some property of plaintexts enables performing query evaluation on ciphertexts. Among them, order-preserving encryption (OPEnc) whose ciphertexts preserve the numerical ordering of their underlying plaintexts has received a lot of attention since it can support efficient query operation on encrypted data such as sorting and range queries using the ordering information. In 2004, Agrawal et al. first proposed the concept of OPEnc. Later, Boldyreva et al. provided the security notions of OPEnc formally and also showed that any immutable OPEnc schemes with ideal security must have the ciphertext length which grows exponentially in the plaintext length. Recently, some ideally-secure OPEnc schemes whose ciphertexts reveal no additional information beyond the order of the underlying plaintexts have been proposed. However, these schemes require large communication and storage complexities.

A new ideally-secure OREncS scheme with shorter ciphertexts is proposed in 2020. Combining it with the domain-extension scheme the new OREncL scheme with shorter ciphertexts under the same security level is obtained ...