\	assumed -		(<i>גע</i> יר)									
1pproach 1: have	a key-distribu	tion center	(KUC)		N N N		VDC	ا ما		0		
				s		between			, I			
					- 1 Y	worts to						, p
		KDC										li to Pj
						- P; cho	soses honce	· '; ~~	, tours	וופר ונו.	to P; an	KUC .
	P. Z	$ \sqrt[3]{\int_{\mathbf{k}_{2}}} $	J _D		- Cr.	- KDC colled C leet" t	samples .	Kij WAR I. I.	gives		1.	0
	, ,	12	1 10		a "tie	bet" 1	$C \leftarrow MC$	Kijenc j K Ll.	(c. c	:4: -1 /	} 16	ri
							;	- Crimaci Chi	1	100, 100,	1	
						t.	; \(\pm \)	CRJ/ENC, R	زن د .	٠١.٠١	2) t to	rj
						, ,	j MIH	- (Kj,MAC,	(c;,1j,	iai, iar,	;)) J	
												and iden
					needed	to bind	. Session	rey Kij	to idea	nnes idi,	(%)	
	V 3											
asic design for								, ,	,			
- Drawback: K	.DC must be t	ally trusted	(knows em	eryone's	Keys) a	id is sing	<u>de</u> point	ot tai	ure (no sessio	in setup i	t kDC
9	oes offline!)											
o develop this, w	e will need	to introduce			a KDC algebra	- number	- theory.					
			some i	abstract	algebra			C		6.11		
			some i	abstract	algebra			atisties	the	following	properties.	
efinition. A grow - Closure:	p consists o	of a set () then g,*(some of together	abstract	an open	ation *	that s	atisfies	the	following	Properties.	
efivition. A grow - Closure: - - Associativity:	p consists of	of a set (then g, *; , g2, g3 E (some of together by the book of the book o	abstract with	$a_{s} = a_{s}$	ation *	that s					
efinition. A grow - Closure: Associativity: - Identity: -	p consists of If gigze 6, For all gi	of a set (1 then g,*; , g2, g3 & (1 an elemen	some of together by EG (abstract with (92* a B such	algebra an oper (a) = (a) that	ation * 1, * 92) * e*9	that s	*e f	پر م <i>ا</i> ا	, ge G	,	
efinition. A grow - Closure: - - Associativity: - - Identity: - - Inverse:	p consists of gigs 6, get 6, get all gight gift for every elements	of a set (then g,*; , g2, g3 E (an element gE	some of together by g, * the e e (6, there	abstract r with (92* g 6 such exists	an oper (a) = (a) that an ele	action $*$ $(1, *, g_e) *$ $e * g_e$ ment g^{-1}	that s 1 93 2 9 = 9 6 6 8	*e fuch the	br all + g*	, ge G 5' = e =	,	
cfinition. A grow - Closure: - Associativity: - Identity: - Inverse: In addition, we	p consists of If gigze 6; For all gi There exists For every el Soup a grow	of a set (then gits, , g2, g3 E (an element gement gE up is comm	some of together by g1 * the e e (6, there whative	abstract r with (92* g B such exists (or ali	algebra an oper (a) = (a) that an ele selon) if	action $*$ $(1, *, g_e) *$ $e * g_e$ ment g^{-1}	that s 1 93 2 9 = 9 6 6 8	*e fuch the	br all + g*	, ge G 5' = e =	,	
efinition. A grow - Closure: - - Associativity: - Identity: - - Inverse:	p consists of If gigze 6; For all gi There exists For every el Soup a grow	of a set (then gits, , g2, g3 E (an element gement gE up is comm	some of together by g1 * the e e (6, there whative	abstract r with (92* g B such exists (or ali	algebra an oper (a) = (a) that an ele selon) if	action $*$ $(1, *, g_e) *$ $e * g_e$ ment g^{-1}	that s 1 93 2 9 = 9 6 6 8	te fach the	€r α/1 + g*	, g & @ g-1 = e = holds:	-5' * g	
Efinition. A grow - Closure: - - Associativity: - Identity: - - Inverse: - - addition, we - Commutativ	p consists of If gigz & G For all gi There exists For every el - soy a grove: For all	of a set (then g,*; , g2, g3 E (an element gement gE up is comm g1, g2 E (some of together by g1 * 6 Co, there whative y1 * 9;	abstract r with (g2* g B such exists (or ab = g2*	algebra an oper (a) = (a) that an ele selon) if	exg ment g the	that s - 93 - 9 = 9 - 6 = 80	* e fuch the	t g* also	ge C gi = e = holds:	= 9" * g	tion
efinition. A grow - Closure: - Associativity: - Identity: - Inverse: In addition, we - Commutative otation: Typically	p consists of If gigze 6. For all gi There exists For every elements Soy a grown e: For all 1, we will use	of a set (then g,*; , g2, g3 E (an element gement gE up is comm g1, g2 E ()	some of together by the context of t	abstract r with (g2* g B such exists (or als 1 = g2*	an oper (a) = (a) that an ele selon) if	extion * exg ment g ¹ the f	that s g3 g = g E G so following	* e fuch the property	for all the g* called all other	ge @ 5' = e = holds: multiplica muire).	= 9" * g active" note We will	tion write
efinition. A grow - Closure: - Associativity: - Identity: - Inverse: In addition, we - Commutative otation: Typically	p consists of If gigz & G For all gi There exists For every el - soy a grove: For all	of a set (then g,*; , g2, g3 E (an element gement gE up is comm g1, g2 E ()	some of together by the context of t	abstract r with (g2* g B such exists (or als 1 = g2*	an oper (a) = (a) that an ele selon) if	extion * exg ment g ¹ the f	that s g3 g = g E G so following	* e fuch the property	for all the g* called all other	ge @ 5' = e = holds: multiplica muire).	= 9" * g active" note We will	tion write
efinition. A grow - Closure: - Associativity: - Identity: - Inverse: In addition, we - Commutative otation: Typically	p consists of If gigze 6. For all gi There exists For every elements Soy a grown e: For all 1, we will use	of a set (then g,*; , g2, g3 E (an element gement gE up is comm g1, g2 E ()	some of together by the context of t	abstract r with (g2* g B such exists (or als 1 = g2*	an oper (a) = (a) that an ele selon) if	extion * exg ment g ¹ the f	that s g3 g = g E G so following	* e fuch the property	for all the g* called all other	ge @ 5' = e = holds: multiplica muire).	= 9" * g active" note We will	tion write
efinition. A grow - Closure: - Associativity: - Identity: - Inverse: In addition, we - Commutativ otation: Typically gx to	p consists of If gigz & Go For all gi There exists For every election E : For all I, we will use denote	of a set (then gits, , g2, g3 E (an element gement gE up is comm g1, g2 E G "" to d g-g-g-g x times	some of together by g, * g. enote the	abstract r with (g2* g B such exists (or ali 2 g2* e gnow usual	algebra an oper (a) = (a) that an ele selian) if g p operation exponenti	extion * exg ment g ¹ the f	that s g3 g = g E G so following	* e fuch the property	for all the g* called all other	ge @ 5' = e = holds: multiplica muire).	= 9" * g active" note We will	tion write
efinition. A grow - Closure: - Associativity: - Identity: - Inverse: In addition, we - Commutativ otation: Typically gx to	p consists of If gigle 6. For all gi There exists For every elements Soy a grown e: For all i, we will use denote	of a set (then gits, , g2, g3 E (an element gement gE up is comm g1, g2 E ("" to d g-g-g-g x times): real no	some of together by a factor of the context of the	abstract r with (gz* g B such exists (or ali = gz* e gnow usual	algebra an oper (a) = (a) that an ele selon) if go operation exponential	extion * exg ment g ¹ the f	that s g3 g = g E G so following	* e fuch the property	for all the g* called all other	ge @ 5' = e = holds: multiplica muire).	= 9" * g active" note We will	tion write
efinition. A grow - Closure: - Associativity: - Identity: - Inverse: In addition, we - Commutativ otation: Typically gx to	p consists of If gigz & Go For all gi There exists For every el soy a grove: For all denote (TR, + (Z, +	of a set (then g,*; , g2, g3 E (an element gement gE up is comm g1, g2 E ("" to d g. g. g X times): real no): integers	some of together by g, * g; enote the g (the moves under	abstract (g2* g B such exists (or ali 2 g2* e gnow usual der ad addition	an oper (a) = (a) that an ele selon) if go operation exponention	extion * exg ment g the f n (unless	that s g3 g = g E G so following explicitly n). We	* e fach the property specified	t g* , also called d other	holds: multiplica muire). note the	= g1 * g ative* note We will multiplice	tion write
efinition. A grow - Closure: - Associativity: - Identity: - Inverse: In addition, we - Commutativ otation: Typically gx to	p consists of If gigz & Go For all gi There exists For every el soy a grove: For all denote (TR, + (Z, +	of a set (then gits, , g2, g3 E (an element gement gE up is comm g1, g2 E ("" to d g-g-g-g x times): real no	some of together by g, * g; enote the g (the moves under	abstract (g2* g B such exists (or ali 2 g2* e gnow usual der ad addition	an oper (a) = (a) that an ele selon) if go operation exponention	extion * exg ment g the f n (unless	that s g3 g = g E G so following explicitly n). We	* e fach the property specified	t g* , also called d other	holds: multiplica muire). note the	= g1 * g ative* note We will multiplice	tion write
efinition. A grow - Closure: - Associativity: - Inverse: In addition, we - Commutativ otation: Typically gx to xamples of groups	p consists of gigg & 6. For all gigg For every elesson a grown a grow	of a set (then gits, , g2, g3 E (an element genent gE up is comm g1, g2 E ("" to d g-g-g-g x times): real no): integers integers printegers	some of together by a factor of the context of the	abstract r with (g2* g B such exists (or ob 2 = g2* e gnow usual der ad addition	algebra an oper (a) = (a) that an ele selon) if go operation exponenti diffion nder ad	extion * 1. * g2) * e * g ment g ⁻¹ The f n (unless id notation dition	that s g3 g = g E G so following explicitly n). We	* e fach the property specified	t g* , also called d other	holds: multiplica muire). note the	= g1 * g ative* note We will multiplice	tion write
Identity: Inverse: In addition, we Commutative Intotation: Typically	p consists of If 9,3,6 G, For all 9, There exists For every elements E soy a grown E : For all denote (TR, + (Zp, + of Z* (o	of a set (then gite , g2, g3 E (an element generat gE up is comm g1, g2 E ("" to d y g g . g X times): real no integers integers re, p is prin	some of together by the e e () g, * g; there extative enote the g (the enote the g (the enote the enote t	abstract r with (92* 9 5 such exists (or ali 2 92* e grow usual der ad addition	algebra an oper (a) = (a) that an ele selian) if groperation exponenti dition nder ad ryptograph	extion * exg exg ment g the f n (unless al notation dition	that s g3 g = g E G so following explicitly n). We	* e fach the property specified	t g* , also called d other	holds: multiplica muire). note the	= g1 * g ative* note We will multiplice	tion write

coefficients

Bezout's identity: For all positive integers $X, y \in \mathbb{Z}$, there exists integers $a, b \in \mathbb{Z}$ such that ax + by = acd(x, y).

Corollary: For prime p, $\mathbb{Z}_p^* = \{1, 2, ..., p-1\}.$

Proof. Take any $\chi \in \{1,2,...,p-1\}$. By Bezout's identity, $\gcd(x,p)=1$ so there exists integers $a,b\in\mathbb{Z}$ where 1=ax+bp. Modulo p, this is ax=1 (mod p) so $a=x^{-1}$ (mod p).

Coefficients a, b in Bezout's identity can be efficiently computed using the extended Euclidean algorithm:

Euclidean algorithm: algorithm for computing gld (a, b) for positive integers a>b:

relies on fact that god (a, b) = god (b, a (mod b):

to see this: take any a > b

L> we can write $\alpha = b \cdot g + r$ where g > 1 is the quotient and $0 \le r \le b$ is the remainder

 \Rightarrow d divides a and b \iff d divides b and \cap \Rightarrow gcd(a,b) = acd(b, \cap) = acd(b, a (mod b))

gives an explicit algorithm for computing god: repeatedly divide:

gcd (60, 27): 60 = 27(2) + 6 [q = 2, r = 6] ~ 9 gcd (60, 27) = gcd (27, 6) 27 = 6(4) + 3 [q = 4, r = 3] ~ 9 gcd (27,6) = gcd (6,3) 6 = 3(2) + 0 [q = 2, r = 0] ~ 7 gcd (6,3) = gcd (3,0) = 3

"rewind" to recover coefficients in Besout's identity:

extended $\begin{cases} 60 = 27(2) + 6 \\ 27 = 6(4) + 3 \end{cases} \Rightarrow 3 = 27 - 6 \cdot 4$ = 27(2) + 6 = 3(2) + 0 = 27(9) + 60(-4)

Iterations readed: O(loga) - i.e., bit-length of the input [worst case inputs: Fibonacci numbers]

Implication: Euclidean algorithm can be used to compute modular inverses (faster algorithms also exist)

```
defined to be the identity element
Definition. A group G is cyclic if there exists a generator g such that G = \{g^{\circ}, g^{\dagger}, ..., g^{|G|-1}\}.

Definition. For an element g \in G, we write \{g\} = \{g^{\circ}, g^{\dagger}, ..., g^{|G|-1}\} to denote the set generated by g (which need not be the
            entire set. The coordinality of (g) is the order of g (i.e., the size of the "subgroup" generated by g)

Consider \mathbb{Z}_7^* = \{1,2,3,4,5,6\}. In this case,
Example. Consider Z7 = {1,2,3,4,5,6}. In this case,
                   \langle 2 \rangle = \{1, 2, 4\} [2 is not a generator of \mathbb{Z}_7^*] ord (2) = 3
                   \langle 3 \rangle = \{1,3,2,6,4,5\} [3 is a generator of \mathbb{Z}_7^*] ord(3) = 6
Lagrange's Theorem. For a group B, and any element g \in G, ord (g) | |G| (the order of g is a divisor of |G|).
           L> For Zp, this means that ord(g) | p-1 for all g ∈ 6
Corollary (Fernat's Theorem): For all x \in \mathbb{Z}_p^*, x^{p-1} = 1 \pmod{p}

Proof. |\mathbb{Z}_p^*| = |\{1,2,...,p-1\}| = p-1
                                                                            for integer k
         By Lagrange's Theorem, ord (x) |p-1| so we can write |p-1|=k \cdot \operatorname{ord}(x) and so |x|^{p-1}=(x^{\operatorname{ord}(x)})^k=1^k=1\pmod p
<u>Implication</u>: Suppose X \in \mathbb{Z}_p^* and we want to compute X^0 \in \mathbb{Z}_p^* for some large integer y \gg p
                      since x^{p-1} = 1 \pmod{p}
                      -> Specifically, the exponents operate modulo the order of the group
    Equivalently: group \langle g \rangle generated by g is isomorphic to the group (\mathbb{Z}_g, +) where g = \operatorname{ord}(g)
                                \langle g \rangle \cong' (\mathbb{Z}_{g}, +)
g^{\chi} \mapsto \chi
Notation: gx denotes g.g....g
           g x denotes (gx) [inverse of group element gx]
          g^{\chi^{-1}} denotes g^{(\chi^{-1})} where \chi^{-1} computed mod ord (g) — need to make sure this inverse exists!
Computing on group elements: In criptography, the groups we typically work with will be large (e.g., 256 or 2024)
    - Size of group element (# bits): ~ log | G| bits (256 bits / 2048 bits)
    - Group operations in Zp*: log p bits per group element
                                      addition of mod p elements: O(log p)
                                      multiplication of mod p values: naively O(log2 p)
                                                                         Karatsuba O(log127)
                                                                         Schönhage - Strassen (GMP library): O(log p log log p log log log p)
                                                                        best algorithm O(log p log log p) [2019]
                                                                                  hot yet processed (> 2 to be faster ... )
                                       exponentiation: using repeated equaring: g, g2, g4, g8, ..., g100 P1, can implement using O(log p)
                                                        multiplications [O(log3 p) with noise multiplication]
                                                           > time/space trade-offs with more precomputed values
                                      division (inversion): typically O(log p) using Euclidean algorithm (can be improved)
```