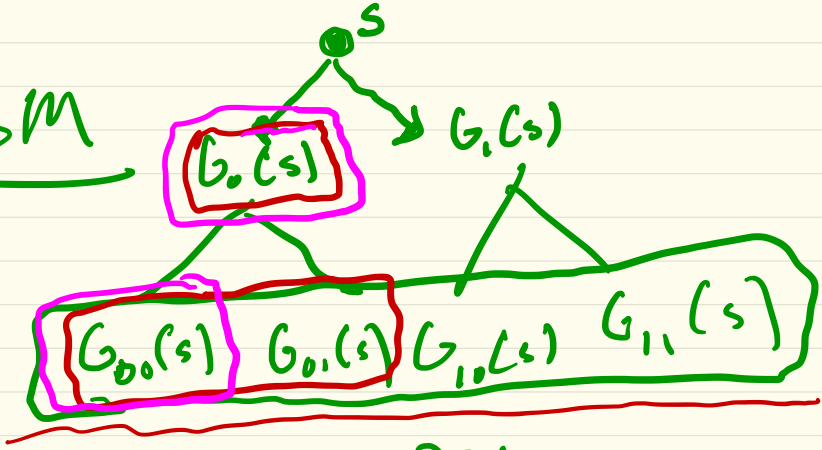
$G: \{0,1\}^n \to \{0,1\}^{2n}$

GGM



If G is PRG
then GGM is PRF.

prefix-free

What about a variable length PRF? It is

prefix-free: Adversary not allowed to query x, y s.t. x is a proper prefix of y.

Variable length tree construction

taste of the proof that GGM is a prefix-free variable length PRF.

$f \xleftarrow{R} \mathrm{Funs}[\{0,1\}^{\leq j}, \mathcal{S}]$

upon receiving a query $x = (a_1, \ldots, a_n) \in \{0,1\}^{\leq \ell}$ from $\mathcal{A}$ do:

if $n < j$ then

then $y \leftarrow f(x)$

else $u \leftarrow (a_1, \ldots, a_j)$, $v \leftarrow (a_{j+1}, \ldots, a_n)$, $y \leftarrow G^*(f(u), v)$

send y to $\mathcal{A}$.

hybrid j

G^* GGM PRF

$f(u)$ label of u

seed for G^*

For $j = 0, \ldots, \ell$, define p_j to be the probability that $\mathcal{A}$ outputs 1 in Hybrid j. As the reader may easily verify, we have

$$\mathrm{PRF}^{\mathrm{pf}}\mathsf{adv}[\mathcal{A}, \tilde{F}] = |p_\ell - p_0|.$$

Next, we define an efficient PRG adversary $\mathcal{B}'$ that attacks the Q-wise parallel composition G' of G, such that

$$\mathrm{PRG}\mathsf{adv}[\mathcal{B}', G'] = \frac{1}{\ell} \cdot |p_\ell - p_0|.$$

Adversary $\mathcal{B}'$ runs as follows:

upon receiving $\vec{r}$ as in (4.32) from its challenger, $\mathcal{B}'$ plays the role of challenger to $\mathcal{A}$, as follows:

$\omega \xleftarrow{R} \{1, \ldots, \ell\}$

initialize an empty associative array $Map : \{0,1\}^* \to \mathbb{Z}_{>0}$

$ctr \leftarrow 0$

upon receiving a query $x = (a_1, \ldots, a_n) \in \{0,1\}^{\leq \ell}$ from $\mathcal{A}$ do:

(*) if $n < \omega$ then

$y \xleftarrow{R} \mathcal{S}$

else

$u \leftarrow (a_1, \ldots, a_{\omega-1}), d \leftarrow a_\omega, v \leftarrow (a_{\omega+1}, \ldots, n)$

if $u \notin \text{Domain}(Map)$ then

$ctr \leftarrow ctr + 1$, $Map[u] \leftarrow ctr$

$p \leftarrow Map[u]$, $y \leftarrow G^*(r_{pd}\, v)$

send y to $\mathcal{A}$.

Finally, $\mathcal{B}'$ outputs whatever $\mathcal{A}$ outputs.

For $b = 0, 1$, let W_b be the event that $\mathcal{B}'$ outputs 1 in Experiment b of Attack Game 4.2 with respect to G'. It is not too hard to see that for any fixed $j = 1, \ldots, \ell$, we have

$$\Pr[W_0 \mid \omega = j] = p_{j-1} \quad \text{and} \quad \Pr[W_1 \mid \omega = j] = p_j.$$

Multi-challenge PRG

$D(t_1, \ldots, t_\ell)$ $\ell = 1$ standard PRG

either $t_i = G(s_i)$ for $s_i \xleftarrow{\$} \{0,1\}^n$ $\forall i$

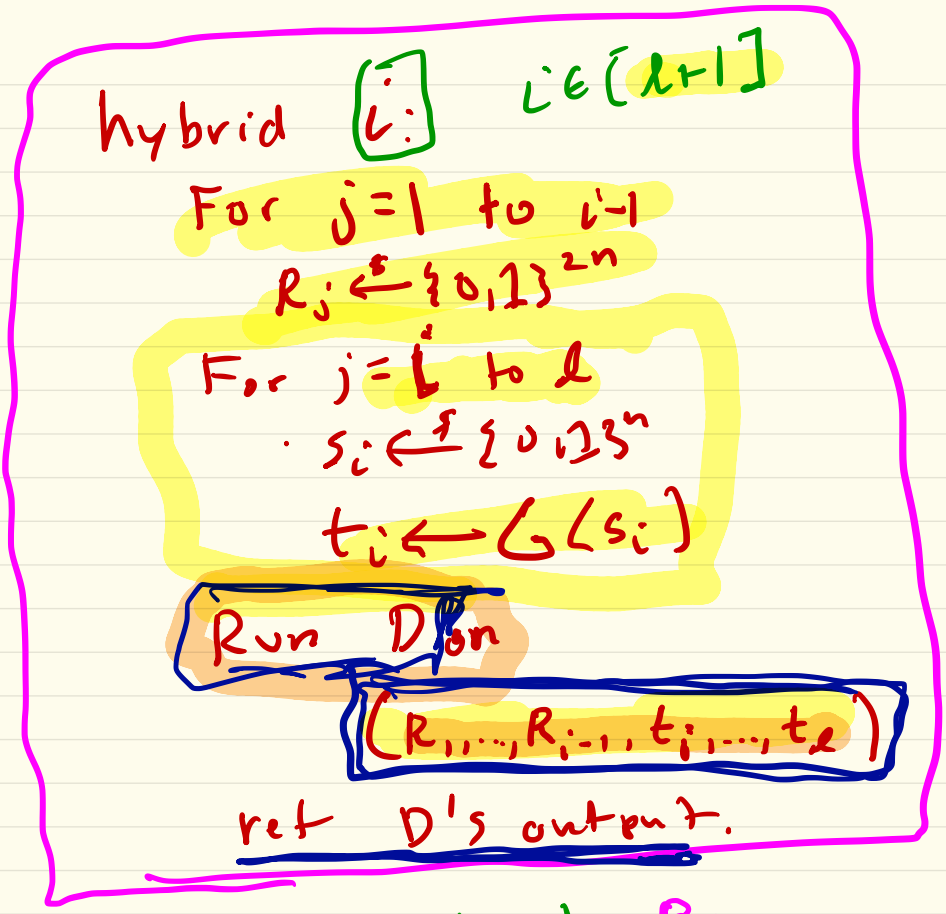
or $t_i \xleftarrow{\$} \{0,1\}^{2n}$ $\forall i$

$G : \{0,1\}^n \to \{0,1\}^{2n}$

WTS single-challenge PRG security $\Rightarrow$ multi-challenge PRG security

how??

Define $\ell+1$ hybrids (aka games) such that in the i-th hybrid $t_1, \ldots, t_{i-1}$ are "real" $t_i, \ldots, t_\ell$ are "random"



Want to construct B_i

s.t. $\Pr[\text{Hybrid } i+1 \Rightarrow 1] - \Pr[\text{Hybrid } i \Rightarrow 1] \leq Adv^{PRG}_{G}(B_i)$

single challenge

$$Adv_G^{mc\text{-}prg}(D) = \Pr[\text{Hybrid } 1 \Rightarrow 1] - \Pr[\text{Hybrid } l+1 \Rightarrow 1]$$

WTS

$$Adv_G^{mc\text{-}prg}(D) \leq \sum_{i=1}^{l+1} Adv_G^{prg}(B_i)$$

Expand!

$$\Pr[\text{Hyb } 1 \Rightarrow 1] - \Pr[\text{Hyb } l+1 \Rightarrow 1]$$

$$= \Pr[\text{Hyb } 1 \Rightarrow 1] - \Pr[\text{Hyb } 2 \Rightarrow 1] + (\Pr[\text{Hyb } 2 \Rightarrow 1] - \Pr[\text{Hyb } 3 \Rightarrow 1]) \cdots$$

Claim. $\exists$ B_i s.t. $\Pr[Hyb_i = 1] - \Pr[Hyb_{i-1} = 1] \leq$ $Adv(B_i)$

single chal. PRG adv.

→ Adversary $B_i(z)$ ← single challenge

For $j = 1$ to $i-1$

$R_j \xleftarrow{R} \{0,1\}^{2n}$

For $j = i+1$ to d

$s_j \xleftarrow{R} \{0,1\}^n$

$T_j \leftarrow G(s_j)$ // $T_j \in \{0,1\}^{2n}$

$d \leftarrow D(R_1, \ldots, R_{i-1}, z, T_{i+1}, \ldots, T_d)$

ret d // $d \in \{0,1\}$

Key Point!

Depending on what z is (i.e. real or random), B_i exactly simulates either Hyb_i or Hyb_{i+1} for D.

$G: \{0,1\}^n \to \{0,1\}^{2n}$

Let D be a distinguisher

single challenge

Exp 1

$s \xleftarrow{R} \{0,1\}^n$

$y \leftarrow G(s)$

ret y

Exp 2

$y \xleftarrow{R} \{0,1\}^{2n}$

ret y.

$$Adv(D) = Pr[D(Exp\,2) \Rightarrow 1] - Pr[D(Exp\,2) \Rightarrow 1]$$

Exp 1

$s_1, \ldots, s_q \xleftarrow{R} \{0,1\}^n$

$y_i \leftarrow G(s_i)\ \forall i \in [q]$

ret $(y_1 \ldots y_q)$

Exp 2

$y_1, \ldots, y_q \xleftarrow{R} \{0,1\}^{2n}$

ret $(y_1, \ldots, y_q)$

length-doubling PRG from PRF

→ $F: \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ is a PRF

→ Define $G: \{0,1\}^n \to \{0,1\}^{2n}$ by:

⟹ $Alg\ G(s)$: // $s \in \{0,1\}^n$

ret $F_s(\langle 0 \rangle) \| F_s(\langle 1 \rangle)$

F can actually be $F: \{0,1\}^n \times \{0,1\}^l \to \{0,1\}^n$ generalized to l

Now run the PRG through the GGM PRF construct.

Claim. You get NMAC w/ block-length l.

alg $NMAC_k(x[1] \ldots x[n])$

$t \leftarrow k$

For $i = 1$ to n

$t \leftarrow F_t(x[i])$

ret t

block-length l

Computational Number Theory

Adam O’Neill

Based on http://cseweb.ucsd.edu/~mihir/cse207/

Secret Key Exchange

History

- Cryptography existed for thousands of years as only symmetric-key.

History

- Cryptography existed for thousands of years as only symmetric-key.
- Nobody thought secret key exchange was possible.

History

- Cryptography existed for thousands of years as only symmetric-key.

- Nobody thought secret key exchange was possible.

- In the 1970's, Diffie and Hellman, and Merkle, proposed the first secret key exchange protocols.

History

- Cryptography existed for thousands of years as only symmetric-key.

- Nobody thought secret key exchange was possible.

- In the 1970's, Diffie and Hellman, and Merkle, proposed the first secret key exchange protocols.

- Public-key (asymmetric) cryptography was born.

History

- Cryptography existed for thousands of years as only symmetric-key.

- Nobody thought secret key exchange was possible.

- In the 1970's, Diffie and Hellman, and Merkle, proposed the first secret key exchange protocols.

- Public-key (asymmetric) cryptography was born.

- Protocols are based on computational group theory and number theory so we first study that.

Some Notation

$\mathbf{Z} = \{\ldots, -2, -1, 0, 1, 2, \ldots\}$

$\mathbf{N} = \{0, 1, 2, \ldots\}$

$\mathbf{Z}_+ = \{1, 2, 3, \ldots\}$

For $a, N \in \mathbf{Z}$ let $\gcd(a, N)$ be the largest $d \in \mathbf{Z}_+$ such that d divides both a and N.

Modular Arithmetic

For $N \in \mathbf{Z}_+$, let

- $\mathbf{Z}_N = \{0, 1, \ldots, N-1\}$
- $\mathbf{Z}_N^* = \{a \in \mathbf{Z}_N \ : \ \gcd(a, N) = 1\}$
- $\varphi(N) = |\mathbf{Z}_N^*|$

Example: $N = 12$

- $\mathbf{Z}_{12} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11\}$
- $\mathbf{Z}_{12}^* =$

Division and mod

$\text{INT-DIV}(a, N)$ returns (q, r) such that

- $a = qN + r$
- $0 \leq r < N$

Refer to q as the quotient and r as the remainder. Then

$$a \bmod N = r \in \mathbf{Z}_N$$

is the remainder when a is divided by N.

Example: $\text{INT-DIV}(17, 3) = (5, 2)$ and $17 \bmod 3 = 2$.

Def: $a \equiv b \pmod N$ if $a \bmod N = b \bmod N$.

Example: $17 \equiv 14 \pmod 3$

Groups

Let G be a non-empty set, and let $\cdot$ be a binary operation on G. This means that for every two points $a, b \in G$, a value $a \cdot b$ is defined.

Example: $G = \mathbf{Z}_{12}^*$ and "$\cdot$" is multiplication modulo 12, meaning

$$a \cdot b = ab \bmod 12$$

Def: We say that G is a *group* if it has four properties called closure, associativity, identity and inverse that we present next.

Fact: If $N \in \mathbf{Z}_+$ then $G = \mathbf{Z}_N^*$ with $a \cdot b = ab \bmod N$ is a group.

Closure

Closure: For every $a, b \in G$ we have $a \cdot b$ is also in G.

Example: $G = \mathbf{Z}_{12}$ with $a \cdot b = ab$ does not have closure because $7 \cdot 5 = 35 \notin \mathbf{Z}_{12}$.

Fact: If $N \in \mathbf{Z}_+$ then $G = \mathbf{Z}_N^*$ with $a \cdot b = ab \bmod N$ satisfies closure, meaning

$$\gcd(a, N) = \gcd(b, N) = 1 \text{ implies } \gcd(ab \bmod N, N) = 1$$

Example: Let $G = \mathbf{Z}_{12}^* = \{1, 5, 7, 11\}$. Then

$$5 \cdot 7 \bmod 12 = 35 \bmod 12 = 11 \in \mathbf{Z}_{12}^*$$

Associativity

Associativity: For every $a, b, c \in G$ we have $(a \cdot b) \cdot c = a \cdot (b \cdot c)$.

Fact: If $N \in \mathbf{Z}_+$ then $G = \mathbf{Z}_N^*$ with $a \cdot b = ab \bmod N$ satisfies associativity, meaning

$$((ab \bmod N)c) \bmod N = (a(bc \bmod N)) \bmod N$$

Example:

$$\begin{aligned}(5 \cdot 7 \bmod 12) \cdot 11 \bmod 12 &= (35 \bmod 12) \cdot 11 \bmod 12 \\ &= 11 \cdot 11 \bmod 12 = 1\end{aligned}$$

$$\begin{aligned}5 \cdot (7 \cdot 11 \bmod 12) \bmod 12 &= 5 \cdot (77 \bmod 12) \bmod 12 \\ &= 5 \cdot 5 \bmod 12 = 1\end{aligned}$$

Exercise: Given an example of a set G and a natural operation $a, b \mapsto a \cdot b$ on G that satisfies closure but *not* associativity.

Identity Element

Identity element: There exists an element $\mathbf{1} \in G$ such that $a \cdot \mathbf{1} = \mathbf{1} \cdot a = a$ for all $a \in G$.

Fact: If $N \in \mathbf{Z}_+$ and $G = \mathbf{Z}_N^*$ with $a \cdot b = ab \bmod N$ then 1 is the identity element because $a \cdot 1 \bmod N = 1 \cdot a \bmod N = a$ for all a.

Inverses

Inverses: For every $a \in G$ there exists a unique $b \in G$ such that $a \cdot b = b \cdot a = \mathbf{1}$.

This b is called the inverse of a and is denoted a^{-1} if G is understood.

Fact: If $N \in \mathbf{Z}_+$ and $G = \mathbf{Z}_N^*$ with $a \cdot b = ab \bmod N$ then $\forall a \in \mathbf{Z}_N^* \quad \exists b \in \mathbf{Z}_N^*$ such that $a \cdot b \bmod N = 1$.

We denote this unique inverse b by $a^{-1} \bmod N$.

Example: $5^{-1} \bmod 12$ is the $b \in \mathbf{Z}_{12}^*$ satisfying $5b \bmod 12 = 1$, so $b =$

Exercises

Let $N \in \mathbf{Z}_+$ and let $G = \mathbf{Z}_N$. Prove that G is a group under the operation $a \cdot b = (a + b) \bmod N$.

Let $n \in \mathbf{Z}_+$ and let $G = \{0, 1\}^n$. Prove that G is a group under the operation $a \cdot b = a \oplus b$.

Let $n \in \mathbf{Z}_+$ and let $G = \{0, 1\}^n$. Prove that G is *not* a group under the operation $a \cdot b = a \wedge b$. (This is bit-wise AND, for example $0110 \wedge 1101 = 0100$.)

Computational Shortcuts

What is $5 \cdot 8 \cdot 10 \cdot 16 \bmod 21$?

Exponentiation

Let G be a group and $a \in G$. We let $a^0 = \mathbf{1}$ be the identity element and for $n \geq 1$, we let

$$a^n = \underbrace{a \cdot a \cdots a}_{n}.$$

Also we let

$$a^{-n} = \underbrace{a^{-1} \cdot a^{-1} \cdots a^{-1}}_{n}.$$

This ensures that for all $i, j \in \mathbf{Z}$,

- $a^{i+j} = a^i \cdot a^j$
- $a^{ij} = (a^i)^j = (a^j)^i$
- $a^{-i} = (a^i)^{-1} = (a^{-1})^i$

Meaning we can manipulate exponents "as usual".

Order

The order of a group G is its size $|G|$, meaning the number of elements in it.

Example: The order of $\mathbf{Z}_{21}^*$ is 12 because

$$\mathbf{Z}_{21}^* = \{1, 2, 4, 5, 8, 10, 11, 13, 16, 17, 19, 20\}$$

Fact: Let G be a group of order m and $a \in G$. Then, $a^m = \mathbf{1}$.

Examples: Modulo 21 we have

- $5^{12} \equiv (5^3)^4 \equiv 20^4 \equiv (-1)^4 \equiv 1$
- $8^{12} \equiv (8^2)^6 \equiv (1)^6 \equiv 1$

Lagrange's Theorem

Simplifying Exponentiation

Corollary: Let G be a group of order m and $a \in G$. Then for any $i \in \mathbf{Z}$,

$$a^i = a^{i \bmod m}.$$

Example: What is 5^{74} mod 21?

Exercises

Evaluate the expressions shown in the first column. Your answer, in the second column, should be a member of the set shown in the third column. In the first case, the inverse refers to the group $\mathbf{Z}_{101}^*$. Don't use any electronic tools; these are designed to be done by hand.

Expression	Value	In
$34^{-1} \bmod 101$		$\mathbf{Z}_{101}^*$
$5^{1602} \bmod 17$		$\mathbf{Z}_{17}^*$
$\lvert\mathbf{Z}_{24}^*\rvert$		$\mathbf{N}$

Running Time

In an algorithms course, the cost of arithmetic is often assumed to be $\mathcal{O}(1)$, because numbers are small. In cryptography numbers are

very, very BIG!

Typical sizes are 2^{512}, 2^{1024}, 2^{2048}.

Numbers are provided to algorithms in binary. The length of a, denoted $|a|$, is the number of bits in the binary encoding of a.

Example: $|7| = 3$ because 7 is 111 in binary.

Running time is measured as a function of the lengths of the inputs.

Algorithms on Numbers

Algorithm	Input	Output	Time
ADD	a, b	$a + b$	linear
MULT	a, b	ab	quadratic
INT-DIV	a, N	q,r	quadratic
MOD	a, N	$a \bmod N$	quadratic
EXT-GCD	a, N	(d, a', N')	quadratic
MOD-INV	$a \in \mathbf{Z}_N^*$, N	$a^{-1} \bmod N$	quadratic
MOD-EXP	a, n, N	$a^n \bmod N$	cubic
EXP_G	a, n	$a^n \in G$	$\mathcal{O}(\lvert n \rvert)$ G-ops

Extended gcd

$\text{EXT-GCD}(a, N) \mapsto (d, a', N')$ such that

$$d = \gcd(a, N) = a \cdot a' + N \cdot N' \ .$$

Example: $\text{EXT-GCD}(12, 20) =$

Extended gcd

$\text{EXT-GCD}(a, N) \mapsto (d, a', N')$ such that

$$d = \gcd(a, N) = a \cdot a' + N \cdot N' \ .$$

Lemma: Let $(q, r) = \text{INT-DIV}(a, N)$. Then, $\gcd(a, N) = \gcd(N, r)$

Alg $\text{EXT-GCD}(a, N)$ // $(a, N) \neq (0, 0)$
if $N = 0$ then return $(a, 1, 0)$
else
  $(q, r) \leftarrow \text{INT-DIV}(a, N)$; $(d, x, y) \leftarrow \text{EXT-GCD}(N, r)$
  $a' \leftarrow y$; $N' \leftarrow x - qy$
  return (d, a', N')

Running time analysis is non-trivial (worst case is Fibonacci numbers) and shows that the time is $\mathcal{O}(|a| \cdot |N|)$. So the extended gcd can be computed in quadratic time.

Modular Inverse

For a, N such that $\gcd(a, N) = 1$, we want to compute $a^{-1} \bmod N$, meaning the unique $a' \in \mathbf{Z}_N^*$ satisfying $aa' \equiv 1 \pmod{N}$.

But if we let $(d, a', N') \leftarrow$ EXT-GCD(a, N) then

$$d = 1 = \gcd(a, N) = a \cdot a' + N \cdot N'$$

But $N \cdot N' \equiv 0 \pmod{N}$ so $aa' \equiv 1 \pmod{N}$

Alg $\mathrm{MOD\text{-}INV}(a, N)$
$(d, a', N') \leftarrow \mathrm{EXT\text{-}GCD}(a, N)$
$\mathrm{return}\ a' \bmod N$

Modular inverse can be computed in quadratic time.

Modular Exponentiation

Let G be a group and $a \in G$. For $n \in \mathbf{N}$, we want to compute $a^n \in G$.

We know that

$$a^n = \underbrace{a \cdot a \cdots a}_{n}$$

Consider:

$y \leftarrow 1$
for $i = 1, \ldots, n$ do $y \leftarrow y \cdot a$
return y

Question: Is this a good algorithm?

Square-And-Mult Example

Suppose the binary length of n is 5, meaning the binary representation of n has the form $b_4 b_3 b_2 b_1 b_0$. Then

$$\begin{aligned} n &= 2^4 b_4 + 2^3 b_3 + 2^2 b_2 + 2^1 b_1 + 2^0 b_0 \\ &= 16 b_4 + 8 b_3 + 4 b_2 + 2 b_1 + b_0 \ . \end{aligned}$$

We want to compute a^n. Our exponentiation algorithm will proceed to compute the values $y_5, y_4, y_3, y_2, y_1, y_0$ in turn, as follows:

$$\begin{array}{lcll} y_5 & = & \mathbf{1} & \\ y_4 & = & y_5^2 \cdot a^{b_4} & = \quad a^{b_4} \\ y_3 & = & y_4^2 \cdot a^{b_3} & = \quad a^{2b_4+b_3} \\ y_2 & = & y_3^2 \cdot a^{b_2} & = \quad a^{4b_4+2b_3+b_2} \\ y_1 & = & y_2^2 \cdot a^{b_1} & = \quad a^{8b_4+4b_3+2b_2+b_1} \\ y_0 & = & y_1^2 \cdot a^{b_0} & = \quad a^{16b_4+8b_3+4b_2+2b_1+b_0} \ . \end{array}$$

Cyclic groups

Let G be a group of order m and let $g \in G$. We let

$$\langle g \rangle = \{ g^i \ : \ i \in \mathbf{Z} \} .$$

Fact: $\langle g \rangle = \{ g^i \ : \ i \in \mathbf{Z}_m \}$

Exercise: Prove the above Fact.

Fact: The size $|\langle g \rangle|$ of the set $\langle g \rangle$ is a divisor of m

Note: $|\langle g \rangle|$ need not equal m!

Definition: $g \in G$ is a generator (or primitive element) of G if $\langle g \rangle = G$, meaning $|\langle g \rangle| = m$.

Definition: G is cyclic if it has a generator, meaning there exists $g \in G$ such that g is a generator of G.

Cyclic groups

Let $G = \mathbf{Z}_{11}^* = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$, which has order $m = 10$.

i	0	1	2	3	4	5	6	7	8	9	10
$2^i \mod 11$	1	2	4	8	5	10	9	7	3	6	1
$5^i \mod 11$	1	5	3	4	9	1	5	3	4	9	1

so

$$\begin{aligned} \langle 2 \rangle &= \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\} \\ \langle 5 \rangle &= \{1, 3, 4, 5, 9\} \end{aligned}$$

- 2 a generator because $\langle 2 \rangle = \mathbf{Z}_{11}^*$.
- 5 is not a generator because $\langle 5 \rangle \neq \mathbf{Z}_{11}^*$.
- $\mathbf{Z}_{11}^*$ is cyclic because it has a generator.

Exercise

Let G be the group $\mathbf{Z}_{10}^*$ under the operation of multiplication modulo 10.

1. List the elements of G

2. What is the order of G?

3. Determine the set $\langle 3 \rangle$

4. Determine the set $\langle 9 \rangle$

5. Is G cyclic? Why or why not?

Discrete log

If $G = \langle g \rangle$ is a cyclic group of order m then for every $a \in G$ there is a unique exponent $i \in \mathbf{Z}_m$ such that $g^i = a$. We call i the discrete logarithm of a to base g and denote it by

$$\mathrm{DLog}_{G,g}(a)$$

The discrete log function is the inverse of the exponentiation function:

$$\begin{aligned} \mathrm{DLog}_{G,g}(g^i) &= i \quad \text{for all } i \in \mathbf{Z}_m \\ g^{\mathrm{DLog}_{G,g}(a)} &= a \quad \text{for all } a \in G. \end{aligned}$$

Discrete log

Let $G = \mathbf{Z}_{11}^* = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$, which is a cyclic group of order $m = 10$. We know that 2 is a generator, so $\mathrm{DLog}_{G,2}(a)$ is the exponent $i \in \mathbf{Z}_{10}$ such that $2^i \bmod 11 = a$.

i	0	1	2	3	4	5	6	7	8	9
$2^i \bmod 11$	1	2	4	8	5	10	9	7	3	6

a	1	2	3	4	5	6	7	8	9	10
$\mathrm{DLog}_{G,2}(a)$										

Finding cyclic groups

Fact 1: Let p be a prime. Then $\mathbf{Z}_p^*$ is cyclic.

Fact 2: Let G be any group whose order $m = |G|$ is a prime number. Then G is cyclic.

Note: $|\mathbf{Z}_p^*| = p - 1$ is not prime, so Fact 2 doesn't imply Fact 1!

Fact 3: If F is a finite field then $F \setminus \{0\}$ is a cyclic group under the multiplicative operation of F.

Computing discrete logs

Let $G = \langle g \rangle$ be a cyclic group of order m with generator $g \in G$.

Input: $X \in G$
Desired Output: $\mathrm{DLog}_{G,g}(X)$

That is, we want x such that $g^x = X$.

for $x = 0, \ldots, m-1$ do
  if $g^x = X$ then return x

Is this a good algorithm? It is

- Correct (always returns the right answer), but
- SLOW!

Run time is $O(m)$ exponentiations, which for $G = \mathbf{Z}_p^*$ is $O(p)$, which is exponential time and prohibitive for large p.

Computing discrete logs

Group	Time to find discrete logarithms
$\mathbf{Z}_p^*$	$e^{1.92(\ln p)^{1/3}(\ln \ln p)^{2/3}}$
EC_p	$\sqrt{p} = e^{\ln(p)/2}$

Here p is a prime and EC_p represents an elliptic curve group of order p.

Note: In the first case the actual running time is $e^{1.92(\ln q)^{1/3}(\ln \ln q)^{2/3}}$ where q is the largest prime factor of $p-1$.

In neither case is a polynomial-time algorithm known.

This (apparent, conjectured) computational intractability of the discrete log problem makes it the basis for cryptographic schemes in which breaking the scheme requires discrete log computation.

Computing discrete logs

In $\mathbf{Z}_p^*$:

$\lvert p \rvert$ **in bits**	**When**
431	2005
530	2007
596	2014

For elliptic curves, current record seems to be for $\lvert p \rvert$ around 113.

Elliptic curve groups

Say we want 80-bits of security, meaning discrete log computation by the best known algorithm should take time 2^{80}. Then

- If we work in $\mathbf{Z}_p^*$ (p a prime) we need to set $|\mathbf{Z}_p^*| = p - 1 \approx 2^{1024}$
- But if we work on an elliptic curve group of prime order p then it suffices to set $p \approx 2^{160}$.

Why? Because

$$e^{1.92(\ln 2^{1024})^{1/3}(\ln \ln 2^{1024})^{2/3}} \approx \sqrt{2^{160}} = 2^{80}$$

But now:

Group Size	Cost of Exponentiation
2^{160}	1
2^{1024}	260

Exponentiation will be 260 times faster in the smaller group!

Discrete log game

Let $G = \langle g \rangle$ be a cyclic group of order m, and A an adversary.

Game $\mathrm{DL}_{G,g}$

procedure Initialize
$x \xleftarrow{\$} \mathbf{Z}_m$; $X \leftarrow g^x$
return X

procedure Finalize(x')
return $(x = x')$

The dl-advantage of A is

$$\mathbf{Adv}^{\mathrm{dl}}_{G,g}(A) = \Pr\left[\mathrm{DL}^{A}_{G,g} \Rightarrow \mathsf{true}\right]$$

Computational Diffie-Hellman

Let $G = \langle g \rangle$ be a cyclic group of order m with generator $g \in G$. The CDH problem is:

Input: $X = g^x \in G$ and $Y = g^y \in G$
Desired Output: $g^{xy} \in G$

This underlies security of the DH Secret Key Exchange Protocol.

Obvious algorithm: $x \leftarrow \mathrm{DLog}_{G,g}(X)$; Return Y^x.

So if one can compute discrete logarithms then one can solve the CDH problem.

The converse is an open question. Potentially, there is a way to quickly solve CDH that avoids computing discrete logarithms. But no such way is known.

CDH Game

Let $G = \langle g \rangle$ be a cyclic group of order m, and A an adversary.

Game $\mathrm{CDH}_{G,g}$

procedure Initialize
$x, y \xleftarrow{\$} \mathbf{Z}_m$
$X \leftarrow g^x$; $Y \leftarrow g^y$
return X, Y

procedure Finalize(Z)
return $(Z = g^{xy})$

The cdh-advantage of A is

$$\mathbf{Adv}^{\mathrm{cdh}}_{G,g}(A) = \Pr\left[\mathrm{CDH}^{A}_{G,g} \Rightarrow \mathsf{true}\right]$$

Building cyclic groups

- Need large groups over which schemes can work

Building cyclic groups

- Need large groups over which schemes can work
- We need generators in these groups

Building cyclic groups

- Need large groups over which schemes can work
- We need generators in these groups
- How to do this efficiently?

Building cyclic groups

To find a suitable prime p and generator g of $\mathbf{Z}_p^*$:

- Pick numbers p at random until p is a prime of the desired form
- Pick elements g from $\mathbf{Z}_p^*$ at random until g is a generator

For this to work we need to know

- How to test if p is prime
- How many numbers in a given range are primes of the desired form
- How to test if g is a generator of $\mathbf{Z}_p^*$ when p is prime
- How many elements of $\mathbf{Z}_p^*$ are generators

Finding primes

Desired: An efficient algorithm that given an integer k returns a prime $p \in \{2^{k-1}, \ldots, 2^k - 1\}$ such that $q = (p-1)/2$ is also prime.

Alg Findprime(k)

do

$p \xleftarrow{\$} \{2^{k-1}, \ldots, 2^k - 1\}$

until (p is prime and $(p-1)/2$ is prime)

return p

- How do we test primality?
- How many iterations do we need to succeed?

Primality testing

Given: integer N
Output: TRUE if N is prime, FALSE otherwise.

for $i = 2, \ldots, \lceil \sqrt{N} \rceil$ do
  if $N \bmod i = 0$ then return false
return true

Density of primes

Let $\pi(N)$ be the number of primes in the range $1, \ldots, N$. So if $p \xleftarrow{\$} \{1, \ldots, N\}$ then

$$\Pr[p \text{ is a prime}] = \frac{\pi(N)}{N}$$

Fact: $\pi(N) \sim \frac{N}{\ln(N)}$

So

$$\Pr[p \text{ is a prime}] \sim \frac{1}{\ln(N)}$$

If $N = 2^{1024}$ this is about $0.001488 \approx 1/1000$.

So the number of iterations taken by our algorithm to find a prime is not too big.

DH Secret Key Exchange

The following are assumed to be public: A large prime p and a generator g of $\mathbf{Z}_p^*$.

Alice		**Bob**
$x \xleftarrow{\$} \mathbf{Z}_{p-1}$; $X \leftarrow g^x \bmod p$		
	$\xrightarrow{X}$	
		$y \xleftarrow{\$} \mathbf{Z}_{p-1}$; $Y \leftarrow g^y \bmod p$
	$\xleftarrow{Y}$	
$K_A \leftarrow Y^x \bmod p$		$K_B \leftarrow X^y \bmod p$

- $Y^x = (g^y)^x = g^{xy} = (g^x)^y = X^y$ modulo p, so $K_A = K_B$
- Adversary is faced with the CDH problem.

DH Secret Key Exchange

- How do we pick a large prime p, and how large is large enough?
- What does it mean for g to be a generator modulo p?
- How do we find a generator modulo p?
- How can Alice quickly compute $x \mapsto g^x \bmod p$?
- How can Bob quickly compute $y \mapsto g^y \bmod p$?
- Why is it hard to compute $(g^x \bmod p, g^y \bmod p) \mapsto g^{xy} \bmod p$?
- ...

Baby-Step Giant-Step

Input: A cyclic group *G* of order *n*, having a generator α and an element β.

Output: A value *x* satisfying $\alpha^x = \beta$.

1. $m \leftarrow \text{Ceiling}(\sqrt{n})$
2. For all *j* where $0 \leq j < m$:
    1. Compute α^j and store the pair (j, α^j) in a table. (See section "In practice")
3. Compute α^{-m}.
4. $\gamma \leftarrow \beta$. (set $\gamma = \beta$)
5. For all *i* where $0 \leq i < m$:
    1. Check to see if γ is the second component (α^j) of any pair in the table.
    2. If so, return $im + j$.
    3. If not, $\gamma \leftarrow \gamma \cdot \alpha^{-m}$.

Testing Primality

Now, let *n* be prime, and odd, with $n > 2$. It follows that $n - 1$ is even and we can write it as $2^s \cdot d$,

$$a^d \equiv 1 \pmod{n}$$

or

$$a^{2^r \cdot d} \equiv -1 \pmod{n}$$

for some $0 \leq r \leq s - 1$.

To show that one of these must be true, recall Fermat's little theorem, that for a prime number n:

$$a^{n-1} \equiv 1 \pmod{n}.$$

The Miller–Rabin primality test is based on the contrapositive of the above claim.

$$a^d \not\equiv 1 \pmod{n}$$

and

$$a^{2^r d} \not\equiv -1 \pmod{n}$$