More efficient crypto? (shorter keys) Public key crypto? (No shared secrets?!)

We'll relax the requirement that our schemes works against computationally bounded adversaries.

Want to protect against "realistic" adversaries. How to model?

We'll use Probabalistic Polynomial Time machines (PPT).

- modeled by Turing Machine w/ a random tape, which halts in $\leq p(|x|)$ steps on input $x \in \{0,1\}^*$

($|x| \triangleq$ length of x)

for some polynomial $p(\cdot)$.

Why randomness? Might actually give more computational power, too! (See BPP = P?)

Example where it doesn't help: primality testing. (See AKS algo.)

Example where it seems to help (polynomial identity testing):

$$M = \begin{pmatrix} f_{11}(x_1,\ldots,x_n) & \cdots & f_{1n}(x_1,\ldots,x_n) \\ \vdots & \ddots & \vdots \\ f_{n1}(x_1,\ldots x_n) & \cdots & f_{nn}(x_1,\ldots,x_n) \end{pmatrix}$$

Question: is $f(x_1,\ldots,x_n) \triangleq \det(M)$

$= 0$ or not?

No subexponential time algo known to deterministically solve! However, if you just plug in some random values for

the x_i and see if the resulting det $= 0$,
you will almost always be right! (det $\neq 0 \Rightarrow f \neq 0$,
det $= 0$ when $f \neq 0$
happens rarely
(see Schwarz lemma...))

Sketch of we want to prove:

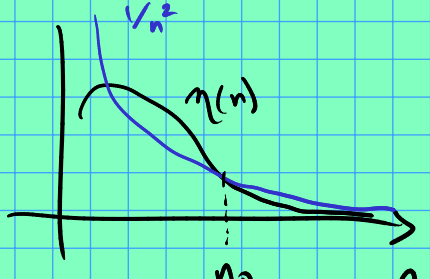
"Violating security of my scheme is hard
for all efficient adversaries."

What does "hard" mean? Adversary has randomness...
they might be able to guess the answer!

Definition: A function $\eta : \mathbb{N} \to \mathbb{N}$ is called negligible
if $\eta(n) = o(n^{-c}) \quad \forall\, c \in \mathbb{N}$.

Put another way: $\forall\, c \in \mathbb{N}$, $\exists\, n_0 \in \mathbb{N}$ $\left(\mathbb{N} = \{1, 2, 3, \dots\}\right)$

s.t. $n > n_0 \Rightarrow \eta(n) < \frac{1}{n^c}$



"Negligible" $\approx$ approaches 0
faster than $1/p(n)$
for any polynomial p
E.g. $\eta(n) = 2^{-n}$.

Better approx. to what we want:

$\Pr[A \in PPT \text{ breaks our security}] < \eta(n)$
for η negligible.

over what distribution?

Usually over random choice of key/instance & coins of A

Hard problems? NP complete? E.g. Traveling Salesman / Hamiltonian Cycles, Boolean satisfiability....

NP ≈ problems w/ answers checkable in polynomial time.

NP = P ?

P = problems solvable in polynomial time.

Even though most researchers think $P \neq NP$, and thus NP-complete problems are hard to solve efficiently, they might not be great for crypto. How come?

NP completeness captures "worst case" hardness.
Such problems might be "easy on average".
No good! Ideally, a random instance should be hard (shouldn't be hard to find keys...)

Definition: One Way Function (easy to compute, hard to invert e.g. Encryption)

$f: \{0,1\}^* \longrightarrow \{0,1\}^*$ is a OWF if

① $\exists$ some poly time algo P_f which computes f:

$P_f(x) = f(x) \quad \forall\ x \in \{0,1\}^*$ (easy to compute)

② $\forall\ A \in PPT, \ \forall\ c \in \mathbb{N}, \ \exists\ l_0 \in \mathbb{N}$ s.t.

$$\Pr_{\substack{x \in_R \{0,1\}^l \\ \text{coins}(A)}}\left[A(y = f(x), 1^l) \in f^{-1}(\{f(x)\})\right] < 1/l^c$$

$\forall\ l > l_0$. (hard to invert)

Note: $1^l = \underbrace{1 \ldots 1}_{l \text{ times}}$. Why give it to A?

Levels playing field between A and the algo computing $f(x)$.

Consider the following $f: \{0,1\}^* \to \{0,1\}^*$ defined as

$$f(x) = [\text{binary rep. of } |x|]$$

E.g. $f(101110) = 111$

Without giving A the 1^ℓ param, this would have been a OWF!

(A's answer is of exponential length, and so would take exp. time just to write down!)

$A(\underbrace{11111\ldots1}_{128 \text{ bits}})$ output 2^{128} bits

Note: OWF definition provides no guarantees for arbitrary x, only for random $x \in \{0,1\}^\ell$.

Also, no guarantee that partial info about input is not revealed!

Say $f: \{0,1\}^* \to \{0,1\}^*$ is a OWF s.t.

$$|f(x)| = |x|.$$

Define $\tilde{f}: \{0,1\}^* \to \{0,1\}^*$ by

$$x = x_0 \| x_1 \longmapsto x_0 \| f(x_1)$$

where $x_0 =$ first $\lfloor |x|/2 \rfloor$ bits of x, $x_1 =$ the rest of x.

Then $\tilde{f}$ also a OWF.

So OWFs seem far from encryption schemes.

However, they do actually imply (symmetric) encryption!

(For the curious: look up the Goldreich-Levin Theorem.)

Candidate OWFs (from number theory)

Integer Factoring? On input $n \in \mathbb{N}$, return a prime factor of n.

if say $n \in_R \{0, \ldots, 2^l - 1\}$, how hard is it to find a factor?

Not very: many numbers will have a small factor

$1/2$ chance of being even...

candidate:

Integer Multiplication

If we restrict to prime numbers of equal length, this is believed to be hard to invert.

I.e., if we choose $n \in_R Primes_l \times Primes_l$

where $Primes_l = \{$prime integers of l bits$\}$.

Easy to compute (forward direction)?

Sure: just use long multiplication ($O(l^2)$ time).

(Aside: can do this faster than $O(l^2)$!
See for example Karatsuba or FFT mult.)

Hard to invert? This is the integer factorization problem above.

No (classical) poly time algo known, but there are efficient quantum algos (see Shor)

If $l \approx 1024$, this problem is thought to be intractable.

Candidate 2: modular exponentiation / discrete logarithms.

First let's recall modular arithmetic facts / defns:

$\mathbb{Z}_n = \{0, 1, 2, \ldots, n-1\}$.

How to do arithmetic with these? Reduce modulo n:

Do normal arithmetic in $\mathbb{Z}$, and keep only the remainder of the result after dividing by n.

E.g. say $n = 10$. Then $2+2 = 4$.

$5+7 = 2$ ($12/10 \to$ remainder is 2)

$3*5 = 5$ ($15/10 \to$ rem is 5).

Note: some nice features of normal arithmetic still work:

negatives / additive inverses: (want to solve $7+X=0$)

$-7 = 3$ since $7+3 = 0 \pmod{10}$.

Also associative / distributive properties still hold.

What about multiplicative inverses?

$7^{-1} = ?$ $\quad 7 \cdot X = 1 \pmod{10}$

$= 3$. since $7 \cdot 3 = 21 \equiv 1 \pmod{10}$

What about 5^{-1}? Has no inverse!! $\pmod{10}$.

$5 \cdot X = 1 \pmod{10}$ has no solution!

For next time: can you see which #'s in $\mathbb{Z}_n$ will have a multiplicative inverse?

(Try to generalize to any value of n.)