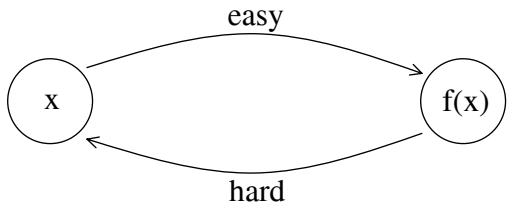
One Way Function

Foundation of Cryptography

Pei-yih Ting
NTOU CS

Contents

Introduction



- Easy: f is a polynomial time computable function
- Hard: for all poly-time probabilistic TM, probability to successfully invert the function is small
- Ex. Possible candidate: multiplication n=p·q is easy,
  factoring n when n is huge is an NP problem
- Finding fundamental intractability assumption: existence of OWF
  - Minimize the number of assumptions
  - Finding some equivalent relations

Negligible Function

- Def: A function $\mu: N \to R$ is negligible
  if for every positive polynomial $p(\cdot)$,
  there exists an n_0 such that for all $n > n_0$
  $\mu(n) < 1/p(n)$
  - Ex. $2^{-\sqrt{n}}$ is negligible
    $n^{-\log n}$ is negligible
  - Negligible functions stays the way when multiplied by any fixed polynomial. i.e.
    $\mu'(n) = p(n)\cdot\mu(n)$ is negligible
    i.e. an event that occurs with negligible probability would be highly unlikely to occur even if we repeated the experiment for polynomially many times

Hard to Invert

✧ For every probabilistic poly-time TM A',
every positive polynomial p(·) and all sufficient large n

$$\Pr\{A'(f(U_n), 1^n) \in f^{-1}f(U_n)\} < 1 / p(n)$$

- U_n is a uniformly distributed r.v. in $\{0, 1\}^n$
- Given TM A', and p(·), finding n_0 such that $\forall\ n > n_0$
- Useless to apply A' for polynomial q(n) times to guess the inverse
- Exclude those function f(·) which can not be inverted by a poly-time A' because output of f(·) is too short such that input of f(·) is exponential in its output

Ex. f(x)=log(x) is not one-way

More Interpretation of 1^n

✧ In Goldreich's book, there are at least two usages of 1^n.
- ✶ The first is in previous slide: the definition of OWF.
- ✶ The second is in computational indistinguishability. i.e.
$\forall$PPT D, $\forall p(\cdot)\ |\Pr\{D(X_n, 1^n)=1\} - \Pr\{D(Y_n, 1^n)=1\}| < 1/p(n)$

✧ The primary goal is to take those simple functions that has output bit length an exponential function of the input bit length back into consideration when discussing PPT, e.g.
- ✶ Consider $\mathbf{f(x) = 2^x}$ (note: if x is 100 bits, $0 < x < 2^{100}$, and $1 < f(x) < 2^{2^{100}}$) Time complexity: $O(|x|^3)$ Space complexity: $O(2^{|x|})$
This function is clearly ruled out as candidate for a PPT D
- ✶ Consider $\mathbf{g(x, 1^n) = f(x) = 2^x}$ (n denotes the output bit length, i.e. $n = 2^{|x|}$) Time complexity: $O((\log_2 n)^3)$ Space complexity: $O(n)$

✧ In principle, we want to consider the complexity as the function of both the input bit length and output bit length.

Hard to Invert

➢ The order of quantifiers is important

$\forall x\ \forall y\ f(x,y) = \forall y\ \forall x\ f(x,y)$

$\exists x\ \exists y\ f(x,y) = \exists y\ \exists x\ f(x,y)$

$\forall x\ \forall y\ \exists z\ f(x,y,z) \neq \forall y\ \exists z\ \forall x\ f(x,y,z)$

For every positive polynomial p(·) and all sufficient large n,
for every probabilistic poly-time TM A',

$$\Pr\{A'(f(U_n), 1^n) \in f^{-1}f(U_n)\} < 1 / p(n)$$

Means: given a p(·), find an n_0, for all TM, it has to satisfy …
this is a more difficult condition to reach

Negation of Quantifier

✧ not ¬

✧ for all (for every) $\forall$

✧ there exist $\exists$

✧ statements f(x, y, z)

✧ $\neg(\forall x, f(x)) = \exists x, \neg f(x)$

ex ¬ all members in the set S is > 0

= at least one member in the set S is ≤ 0

✧ $\neg(\exists x, f(x)) = \nexists x, f(x) = \forall x, \neg f(x)$

ex ¬ at least one member in the set S is ≤ 0

= there does not exist any member in the set S which is ≤ 0

= all members in the set S is > 0

Strong One-Way Function

✧ f(·) is a strong one-way function if

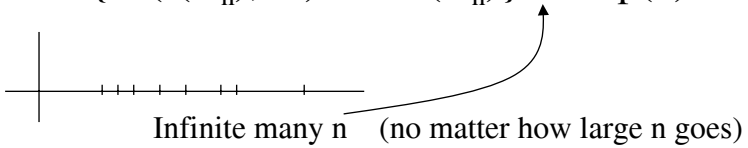
① f is poly-time computable
② f is hard to invert

✧ What does it mean if f(·) is NOT a strong one-way function

∃ a probabilistic poly-time TM A'

∃ positive polynomial p(·) and infinite many n

$$\Pr\{A'(f(U_n), 1^n) \in f^{-1}f(U_n)\} \geq 1 / p(n)$$



Weak One-Way Function

✧ Slightly hard to invert

∃ positive polynomial p(·), s.t. ∀ PPT algorithm A'

and all sufficiently large n

$$\Pr\{A'(f(U_n), 1^n) \notin f^{-1}f(U_n)\} > 1 / p(n)$$

☺ **There is at least slight chances that f(·) can not be inverted**

☺ $\Pr\{A'(f(U_n), 1^n) \in f^{-1}f(U_n)\} \leq 1-1/p(n)$ … successfully inverted probability is bounded from above

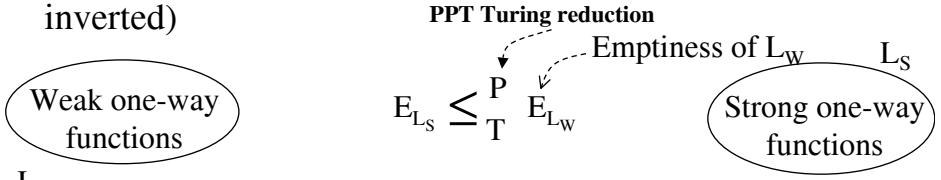
✧ ε-oneway function: consider inversion failures

✶ Weak one-way function is 1/p(n)-one-way

✶ Strong one-way function is (1-1/p(n))-one-way

Weak OWF exists ⇒ Strong OWF exists

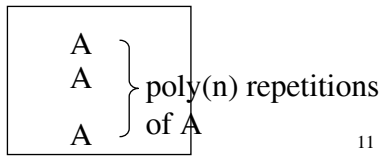
✧ If weak one-way function exists (a particular function can not be inverted) then strong one-way function exists (another function that has a specific form can not be inverted)



Then using A construct another B, B can decide whether a function g(·) of a particular form, ex $g(x)=f(x_1)f(x_2)\cdots f(x_{t(n)})$ is a strong one way function

⇐

If A can decide whether a weak one-way function exists



concatenation

Weak OWF exists ⇒ Strong OWF exists

✧ Because we are still searching for REAL one-way functions, this theorem tells us that we only need to look for a weak one-way function.

✧ Intuition:

string concatenation

$$g(x_1x_2\cdots x_{t(n)}) = f(x_1)\, f(x_2)\cdots f(x_{t(n)})$$

g(·) must be harder to invert than f(·), but how hard is it?

✧ If we assume that inverting $f(x_i)$'s are independent, for each weak OWF $f(x_i)$, the successful inversion probability is bounded above by 1-1/p(n), the probability for successfully inverting $g(x_1x_2\cdots x_n)$ is, therefore, bounded above by $(1-1/p(n))^{t(n)} \rightarrow e^{-n}$ if $t(n)=n\cdot p(n)$

where $\lim_{n\to\infty}(1-\frac{1}{n})^n = e^{-1}$

Looks good, but is the assumption true?

Weak OWF exists ⇒ Strong OWF exists

- Are "inverting $f(x_i)$'s independent"?
  - $f(\cdot)$ is a function with finite domain and range
  - When an algorithm A' tries to invert $f(x_i)$, it must try to probe into the mapping structures from x_i to $f(x_i)$.
  - The more structures A' knows about $f(\cdot)$, the easier A' tries to invert another $f(x_j)$.
  - Sometimes, you might think that if A' fails to invert $f(x_i)$ for many many times, then it must knows a lot about $f(\cdot)$ in its course. Therefore, A' might have a better chance to successfully invert another $f(x_j)$.
- "Inverting $f(x_i)$'s" look not independent, but how bad is it? Are they heavily dependent such that after you invert successfully several $f(x_i)$'s you can easily invert all others?

Weak OWF exists ⇒ Strong OWF exists

- Actually, it is not that pessimistic.
- The plan for proving: again reduction argument, or put it in another way, 'prove by contradiction'
  - Find a construction of $g(\cdot)$ from Weak OWF $f(\cdot)$
  - Assume that $g(\cdot)$ is not a strong OWF, then there exists a PPT B'
  - Using B' polynomial times, construct another PPT algorithm A' to invert $f(\cdot)$
  - Therefore, $f(\cdot)$ is not a weak OWF, contradiction
  - The construction is valid.¶
- PPT algorithm A' (for inverting $f(\cdot)$):

  Input y

  for i=1 to t(n) do

  ① select $x_1, x_2, \ldots x_{t(n)} \in \{0,1\}^n$ randomly, calculate $f(x_1)\ldots f(x_{t(n)})$

  ② compute $z_1, z_2, \ldots z_{t(n)}=B'(f(x_1)\ldots y\ldots f(x_{t(n)})$

  ③ if $y=f(z_i)$ stop and declare success

  repeat the above loop for a(n) times

  replace $f(x_i)$

Weak OWF exists ⇒ Strong OWF exists

- Works remaining: $g() = f()\ f()\ \ldots\ f()$
  - Determine a(n) and t(n), such that
    - given B' can invert the strong one-way function $g(\cdot)$, i.e.

      $\exists$ PPT B', $\exists$ poly $q(\cdot)$ and infinite many n

      $$\Pr\{B'(g(U_n), 1^n) \in g^{-1}g(U_n)\} \geq 1 / q(n)$$
    - Prove that $f(\cdot)$ is not weak one-way, i.e.

      $\forall$ positive polynomial $p(\cdot)$, s.t. $\exists$ PPT TM A' and infinite many n

      $$\Pr\{A'(f(U_n), 1^n) \notin f^{-1}f(U_n)\} \leq 1 / p(n)$$
  - This is not trivial. We shall prove it later.
  - Amplification of computational difficulty is much more involved than amplification of an analogous probabilistic event

One Way Function Exists ⇒ P ≠ NP

- Some more implications about the "OWF exists" intractability assumption
  - This implies that an OWF can form a language that is in NP but not in P.
- Proof:

  Plan: ❶ Given $f(\cdot)$ an OWF

  ❷ Define a specific language $L_f \in$ NP but L_f is clearly $\notin$ P based on the fact that $f(\cdot)$ is difficult to invert

  ❸ If we assume P=NP then $L_f \in$ P, and we can find out a poly-time TM M_f that can invert $f(\cdot)$, i.e. $f(\cdot)$ is not a OWF

  contradiction ¶

One Way Function Exists $\Rightarrow$ P $\neq$ NP

- Consider the decision problem:

  $L_f = \{y \mid \exists\ x \text{ s.t. } f(x) = y, \text{ where } x, y \in \{0,1\}^n\}$

  - This set is the range of a OWF $f(\cdot)$
  - This problem is clearly in NP. The witness is x. Given the witness x of a corresponding y, we can verify the membership of y by testing whether $f(x) = y$ in polynomial time.
    i.e. $\exists$ an NTM, input y, decides whether $\exists$ x s.t. $f(x) = y$ (i.e $y \in L_f$)
    this is definitely true since given y, $\exists$ an NTM that can either guess $x = f^{-1}(y)$ or try out all possible solutions
  - Is this problem not in P given the assumption that f is a OWF??? Looks like true!! The definition of OWF says that there is no poly-time PPT algorithm that can calculate x given an arbitrary y.
- Consider a similar language

  $L_f' = \{(y, b) \mid \exists\ x \le b \text{ s.t. } f(x) = y\}$

One Way Function Exists $\Rightarrow$ P $\neq$ NP

  - This set is the range of $f(\cdot)$ that is not larger than b
  - This problem is clearly in NP. The witness is again x. $L_f' \in$ NP
  - Is L_f' not in P given $f(\cdot)$ a one way function ???
- Assume NP=P then $L_f' \in$ P, i.e. $\exists$ a poly-time TM $M_f'(y,b)$ that decides L_f'
  - Program Invert(y) (constructed using $M_f'(y,b)$)

    given y
    { guess an upper bound of x, say B
    { if $M_f'(y,B)=1$, try B/2 else try 2B

    the reduction
  - This program is a binary search-like algorithm, if the space for x is $\{0, 1\}^{p(n)}$, it is going to stop in p(n) steps
  - Therefore, if L_f' is in P then $f(\cdot)$ is not a OWF ¶
  - Note: in the above we consider the one-way function $f(\cdot)$ where the input length and output length are close (the length preservative OWF or the length regular OWF)

More Implications

- If one-way function exists, hardcore predicate exists
- If one-way function exists, pseudo-random number generator (PRNG) exists
- If one-way function exists, pseudo-random function (PRF) exists
- If one-way function exists, secure bit commitment exists
- If one-way function exists, every NP problem has a ZKA.

Poly-time Enumerable Set

- Sometimes consider OWFs with domains subset of $\{0,1\}^*$

  ex. $\cup_{n \in I} \{0,1\}^{\ell(n)}$ where $\ell(n)$ is some polynomial
- Def:
  - Successor of n in the set I: let $I \subseteq$ Í , $s_I(n) = \min\{i \in I: i > n\}$
    - $s_I(n)$ is the smallest integer that is both greater than n and in the set I
  - A set $I \subseteq$ Í is called polynomial-time-enumerable if

    $\exists$ an algorithm that on input n halts within poly(n) steps and output $1^{s_I(n)}$

    The unary output forces s_I to be polynomially bounded; i.e., $s_I(n) \le$ poly(n)
  - Also define $\ell_I(m) = \max\{i \in I: i \le m\}$, the largest lower bound of m in I

OWF for Poly-time Enumerable Set

- $f(\cdot)$ is strongly one-way on lengths in I if
  - ① f is polynomial-time-computable and
  - ② f is hard to invert over $n \in I$

    For every probabilistic poly-time algorithm A',

    every positive polynomial $p(\cdot)$ and all sufficient large n in I

    $\Pr\{A'(f(U_n), 1^n) \in f^{-1}f(U_n)\} < 1 / p(n)$
- Ordinary OWF can be viewed as being one-way on lengths in Í
- OWFs on lengths in any poly-time-enumerable set can be easily transformed into ordinary one-way functions
  - ⋆ $f(\cdot)$ is strongly one-way on lengths in I
  - ⋆ Construct $g:\{0,1\}^* \to \{0,1\}^*$, $g(x) = f(x')$ where x' is the longest prefix of x with length in I
  - ⋆ Can also construct a length preserving OWF $g'(\cdot)$ from a length preserving OWF $f(\cdot)$ in I, $g':\{0,1\}^* \to \{0,1\}^*$, $g'(x) = f(x')x''$ where $x=x'x''$ and x' is the longest prefix of x with $|x'|$ in I

"Existence of OWF in I" is equivalent to "Existence of OWF"

- **Proposition 2.2.3**: Let I be a poly-time enumerable set, and $f(\cdot)$ be a strong OWF in I then $g(x)=f(x')$ is strongly one-way, where x' is the longest prefix of x with $|x'| \in I$

proof:

① g can be computed in polynomial time:

for any x, finding $\ell_I(|x|)$ is poly-time since I is poly-time enumerable by testing($|x|$-1,$|x|$-2… to see if $s_I(|x|-i)=|x|$)

applying $f(\cdot)$ on x' can be done in poly-time

$s_I(n) = \min\{i \in I: i>n\}$
$\ell_I(m) = \max\{i \in I: i \le m\}$

② assume $g(\cdot)$ is not a strong OWF, i.e. ∃ PPT B' that can invert $g(\cdot)$

construct an algorithm A' that can invert $f(x) = y \in f(U_n)$, $n \in I$

given $y \in f(U_n)$, 1^n, $n \in I$ <------ assume we know the length of x

A' computes $s_I(n)$

for any i, $0 \le i \le s_I(n)-n-1$ {

apply $B'(y, 1^{n+i}) = z_i$ and verify $g(z_i) = y$

if success, output n-bit prefix of z_i

}

"∃ OWF in I" ≡ "∃ OWF" (2/3)

To prove that algorithm A' can invert $f(x) = y \in f(U_n)$, $n \in I$, with non-negligible probabilities, we have to lower bound the following probability

$$\Pr\{A'(f(U_n), 1^n) \in f^{-1}f(U_n)\} \ge \Pr\{B'(g(U_m), 1^m) \in g^{-1}g(U_m)\}$$

for any m such that $n=\ell_I(m) \in I$

i.e. for arbitrary length n string x, $n \in I$, A'(f(x), 1^n) will calculate B'(f(x), 1^t), for all t in the range $n \le t \le s_I(n)-1$, $s_I(n)-n$ times, which includes B'(f(x), 1^m)

for a particular t, $g(xx'') = f(x)$, $\forall\, x'' \in \{0,1\}^{t-|x|}$, if B' successfully inverts any one of g(xx'') then A' succeds.

"∃ OWF in I" ≡ "∃ OWF" (3/3)

By assumption, g is not strongly one-way, i.e.

∃ PPT B', ∃ $p(\cdot)$ for infinitely many m's (let the set of m's be M)

$\Pr\{B'(g(U_m), 1^m) \in g^{-1}g(U_m)\} \ge 1/p(m)$ $M \subseteq$ Í

we need to prove that

∃ $p'(\cdot)$ for infinitely many n's, $n \in M' \subseteq I$,

$\Pr\{A'(f(U_n), 1^n) \in f^{-1}f(U_n)\} \ge 1/p'(n)$

the problem is $m \in M$ but $n \in M'$

① for all $m \in M$, $n=\ell_I(m) \in I$, there exists $q(\cdot)$ s.t. $m<q(\ell_I(m))=q(n)$

since I is poly-time enumerable ⇒ $\forall n \in I$, $s_I(n)<poly(n)$ ⇒

$\forall m \in M$, $m<s_I(\ell_I(m))<poly(\ell_I(m))=q(\ell_I(m))$

therefore, $\Pr\{A'(f(U_n),1^n) \in f^{-1}f(U_n)\} \ge \Pr\{B'(g(U_m),1^m) \in g^{-1}g(U_m)\}$

$\ge 1/p(m) > 1/p(q(n)) = 1/p'(n)$

② $S \triangleq \{\ell_I(m); m \in M\}$ Is |S| infinite? (infinite many n's?)

proof: if |S| is finite, there exists an upper bound $n^*=\ell_I(m^*)$, also from ①, $m^* < q(n^*)$, therefore, |M| is also finite, contradiction

① and ② imply that f is not a strong one-way function.

Length-Regular/Preserving OWFs

- If $f(\cdot)$ is a strong one-way function, then a length-regular function $g(\cdot)$ constructed from $f(\cdot)$ is also a strong one-way function
  - A function f is length-regular if for every $x, y \in \{0,1\}^*$, if $|x|=|y|$ then $|f(x)|=|f(y)|$
  - $g(x) = f(x)\ 1\ 0^{p(|x|)-|f(x)|}$, where $|g(x)| = p(|x|)+1$

- Also a length-preserving strong one way function h can be constructed from a strong one-way function f
  - A function h is length-preserving if for every $x \in \{0,1\}^*$, $|h(x)|=|x|$
  - $h(x) = h(x'x'') = g(x') = f(x')\ 1\ 0^{p(|x'|)-|f(x')|}$, where x' is chosen s. t. $|h(x)| = p(|x'|)+1=|x|$ and x'' is the remaining portion of x

Length-Regular OWFs

- proof:
  - ① g can be computed in polynomial time
  - ② assume $g(\cdot)$ is not a strong OWF, i.e. $\exists$ PPT B' that can invert $g(\cdot)$
    construct an algorithm A' that can invert $f(x)=y \in f(U_n)$, $n \in$ Í
    given $y \in f(U_n)$, 1^n, $n \in$ Í is a possible length for x, $p(n) \geq |y|$
    for any n {
    apply $B'(y10^{p(n)-|y|}) = x$ and verify $g(x)=y10^{p(n)-|y|}$ or $f(x)=y$
    if success, output x
    }

$y10^{p(n')-|y|}$

$$\Pr\{A'(f(U_n), 1^n) \in f^{-1}f(U_n)\} = \sum_{n' \in T} \Pr\{B'(g(U_{n'}), 1^{n'}) \in g^{-1}g(U_{n'})\}$$

$$\geq \Pr\{B'(g(U_m), 1^m) \in g^{-1}g(U_m)\} \quad (m \in T)$$

$\exists$ B', $\exists$ $p(\cdot)$, for infinite many m's, let m's form the set M,
$\Pr\{B'(g(U_m), 1^m) \in g^{-1}g(U_m)\} > 1/p(m)$
For the same set of M, $\Pr\{A'(f(U_n), 1^n) \in f^{-1}f(U_n)\} > 1/p(n)$

Weak OWF exists ⇒ Strong OWF exists

$$g(x_1x_2 \cdots x_{t(n)}) = f(x_1)\ f(x_2) \cdots f(x_{t(n)})$$

where $|x_1|=|x_2|=\cdots|x_{t(n)}|=n$, $t(n)=n \cdot p(n)$

pf: by contradiction

① if g is not strongly one way,
$\exists$ B', $\exists$ $q(\cdot)$, for infinite many m's, let m's form the set M',
$\Pr\{B'(g(U_m)) \in g^{-1}g(U_m)\} > 1/q(m)$
let N' denote the infinite set of n's for which $n^2 \cdot p(n) \in M'$

② using B', we construct a PPT A' for inverting f. A' applies the following procedure I on the input y for a(|y|) times, where $a(n)=2n^2 \cdot p(n) \cdot q(n^2 \cdot p(n)))$

procedure I: Input y, let n=|y|
for i=1 to t(n) do

replace $f(x_i)$

- ❶ select $x_1, x_2, \ldots x_{t(n)} \in \{0,1\}^n$ randomly, calculate $f(x_1)\ldots f(x_{t(n)})$
- ❷ compute $(z_1, z_2, \ldots z_{t(n)})=B'(f(x_1),\ldots,y,\ldots,f(x_{t(n)}))$
- ❸ if $y=f(z_i)$, stop and declare success

W. OWF exists ⇒ S. OWF exists (2/4)

Now present a lower bound on the success probability of algorithm A'

③ define a set S_n: contains all n-bit strings on which I succeeds with noticeable probability, i.e. $S_n=\{x:|x|=n, \Pr\{I(f(x)) \in f^{-1}f(x)\} > n/a(n)\}$

④ $\forall x \in S_n$, $\Pr\{A'(f(x)) \in f^{-1}f(x)\} > 1-2^{-n}$ (prob. increases with repetitions)

pf: $\Pr\{A'(f(x)) \notin f^{-1}f(x)\} < (1-n/a(n))^{a(n)} < e^{-n} < 2^{-n}$

failed for all a(n) repetitions

⑤ $\forall n \in N'$, $|S_n| > (1-\frac{1}{2p(n)}) \cdot 2^n$

pf: assume, to the contrary, that $|S_n| \leq (1-\frac{1}{2p(n)}) \cdot 2^n$ ❁

assumption in ①

let $s(n)=\Pr\{B'(g(U_{n^2p(n)})) \in g^{-1}g(U_{n^2p(n)})\} > 1/q(n^2p(n))$

denote $U_{n^2p(n)} = U_n^{(1)} U_n^{(2)} \cdots U_n^{(n \cdot p(n))}$, $U_n^{(i)}$ are indep. n-bit uniform r.v.

$s(n) = s_1(n) + s_2(n)$

$s_1(n) = \Pr\{B'(g(U_{n^2p(n)})) \in g^{-1}g(U_{n^2p(n)}) \wedge (\exists i \text{ s.t. } U_n^{(i)} \notin S_n)\}$

$s_2(n) = \Pr\{B'(g(U_{n^2p(n)})) \in g^{-1}g(U_{n^2p(n)}) \wedge (\forall i\ U_n^{(i)} \in S_n)\}$

W. OWF exists ⇒ S. OWF exists (3/4)

❶ for every $x\in\{0,1\}^n$ and every $1 \le i \le n\cdot p(n)$

$$\Pr\{I(f(x))\in f^{-1}f(x)\}\ge\Pr\{B'(g(U_{n^2p(n)}))\in g^{-1}g(U_{n^2p(n)}) \mid U_n^{(i)}=x\}$$

$$s_1(n) = \Pr_{n\cdot p(n)}\{\exists i \text{ s.t. } (B'(g(U_{n^2p(n)}))\in g^{-1}g(U_{n^2p(n)})) \wedge (U_n^{(i)}\notin S_n)\}$$

$$\le \sum_{i=1}^{n\cdot p(n)} \Pr\{(B'(g(U_{n^2p(n)}))\in g^{-1}g(U_{n^2p(n)})) \wedge (U_n^{(i)}\notin S_n)\}$$

$$= \sum_{i=1}^{n\cdot p(n)} \sum_{x\notin S_n} \Pr\{(B'(g(U_{n^2p(n)}))\in g^{-1}g(U_{n^2p(n)})) \wedge (U_n^{(i)}=x)\}$$

$$= \sum_{n\cdot p(n)} \sum_{x\notin S_n} \left(\Pr\{U_n^{(i)}=x\}\cdot\Pr\{B'(g(U_{n^2p(n)}))\in g^{-1}g(U_{n^2p(n)})|U_n^{(i)}=x\}\right)$$

$$\le \sum_{i=1}^{n\cdot p(n)} \max_{x\notin S_n} \Pr\{B'(g(U_{n^2p(n)}))\in g^{-1}g(U_{n^2p(n)})|U_n^{(i)}=x\}$$

assumption ❁

$$\le \sum_{i=1}^{n\cdot p(n)} \max_{x\notin S_n} \Pr\{I(f(x))\in f^{-1}f(x)\} \le n\cdot p(n)\cdot\frac{n}{a(n)} = \frac{n^2\cdot p(n)}{a(n)}$$

❷ $s_2(n) \le \Pr\{\forall i: U_n^{(i)}\in S_n\} \le (1-\frac{1}{2p(n)})^{n\cdot p(n)} < e^{-n/2} < \frac{n^2\cdot p(n)}{a(n)}$

RHS is a polynomial

Therefore, $s_1(n) + s_2(n) < \frac{2n^2\cdot p(n)}{a(n)} = \frac{1}{q(n^2\cdot p(n))}$ contradiction

W. OWF exists ⇒ S. OWF exists (4/4)

⑥ $\Pr\{A'(f(U_n))\in f^{-1}f(U_n)\} \ge \Pr\{A'(f(U_n))\in f^{-1}f(U_n) \wedge U_n\in S_n\}$

$$= \Pr\{U_n\in S_n\}\cdot\Pr\{A'(f(U_n))\in f^{-1}f(U_n) \mid U_n\in S_n\}$$

$$\ge (1-\frac{1}{2p(n)})\cdot(1-2^{-n}) > 1-\frac{1}{p(n)}$$

i.e. f(·) is not weakly one-way ¶

- $g(x_1x_2\cdots x_{t(n)}) = f(x_1)\,f(x_2)\cdots f(x_{t(n)})$ is not an efficient construction of strong OWF from weak OWF
  why not using $g(x) = f(f(\cdots f(x)\cdot\cdot))$??
  random walk on expander graph
  see also section 2.6 efficient amplification of OWF

Illustrating Example (1/4)

Consider $g(x_1\,x_2) = f(x_1)\,f(x_2)$ intuitively g is a stronger OWF than f

Let f(·) be 1/3-oneway, i.e.∀A', suff. large n, Pr{invert failure}>1/3

What kind of oneway function will g(·) be?

Naïve assumption: B' invert g indep. on $f(x_1)$ and $f(x_2)$, therefore,

$\Pr\{B' \text{ success}\} \le (2/3)^2 = 4/9$

i.e. $\Pr\{B' \text{ failure}\} > 1-(2/3)^2 = 5/9$

(in fact Pr{B' failure} > some constant slightly less than 5/9)

Let's prove that g is a 0.55-oneway (∀B', suff. large n, Pr{invert failure}>0.55)

proof by contradiction:

assume g is not 0.55-oneway, i.e.

∃ B', infinite many n, Pr{invert failure}≤0.55 (Pr{success}>0.45)

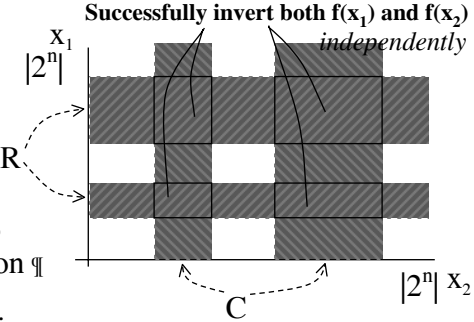
want to show that f is not 1/3 oneway, i.e.

∃ A', infinite many n, Pr{invert failure}≤1/3 (Pr{success}>2/3)

Illustrating Example (2/4)

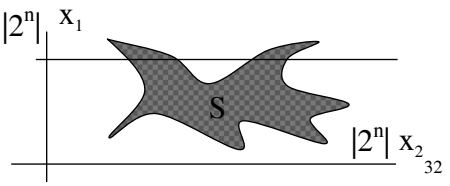
Case ① naïve assumption:

assume B' is deterministic
consider a specific n
assume $|R|\cdot|C| > 0.45\cdot(2^n)^2$
because |R| = |C|
⇒ $|R|>0.6708\cdot2^n>2/3\cdot2^n$
i.e. Pr{A' success}>2/3
contradiction ¶



Case ② in general

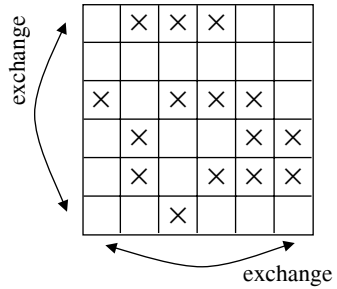
good rows: more than 0.1% 1's in a row
good columns: more than 0.1% 1's in a column
Because Pr{B' invert g successfully}>0.45,
there must be at least $d\cdot2^n$ good rows and $d\cdot2^n$ good columns



Illustrating Example (3/4)

To find an upper bound on |S|, let's consider a simple example:

a row with at least $0.001 \cdot 2^n$ '×' is a good row
a column with at least $0.001 \cdot 2^n$ '×' is a good column



Let S denotes those ×'s, i.e. the region that B' invert successfully

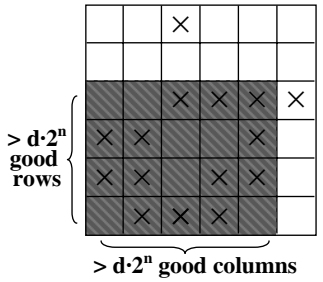
$|S| < d \cdot 2^n \cdot d \cdot 2^n + 2 \cdot (1-d) \cdot 2^n \cdot 0.001 \cdot 2^n$

$\Rightarrow 0.45 < d^2 - 0.002\, d + 0.002$

$\Rightarrow d^2 - 0.002\, d - 0.448 > 0$

$\Rightarrow (d - 0.6703)(d + 0.6683) > 0$

$\Rightarrow d > 0.6703$



Illustrating Example (4/4)

With such an algorithm B', construct an algorithm A' to invert f(·):

Given $y \in f(U_n)$
for (i=0; i<10000; i++)
  select x_2 uniformly in $\{0,1\}^n$
  $B'(y\ f(x_2)) \rightarrow (x'\ x'')$ where $x', x'' \in \{0,1\}^n$
  verify if y = f(x'), output x'

Pr{A' invert y=f(x) successfully}:
  for a good x, Pr{B'(y, $f(x_2)$) successfully invert y} > 0.001
$\Rightarrow$ Pr{A' invert y=f(x) successfully | x is good} = $1 - (1-0.001)^{10000}$
  > 0.999955
$\Rightarrow$ Pr{A' invert y=f(x) successfully} > 0.6703 · 0.999955 > 2/3
  f(·) is not 1/3-oneway contradiction ¶

Collection of OWF

- Previous definition for OWF is for abstract discussion, the function operates on infinite domains i.e. $\{0,1\}^*$
- For describing many natural candidates of OWFs, we consider infinite collections of functions each operating on a ***finite domain***

  - An infinite set of indices: $\hat{I}$ (dense subset of all stings)
  - A corresponding set of finite functions, $\{f_i\}_{i \in \hat{I}}$
  - For each $i \in \hat{I}$, the domain of f_i, denoted D_i, is a finite set
  - An efficient function-evaluation algorithm F, on input $i \in \hat{I}$ and $x \in D_i$ returns $f_i(x)$
  - An efficient algorithm I, on input 1^n, randomly selects a poly(n)-bit-long index $i \in \hat{I}$ specifying $f_i(\cdot)$ and D_i
  - An efficient algorithm D, on input $i \in \hat{I}$, randomly selects $x \in D_i$

➢ This formulation of a collection of functions is also useful for the presentation of trapdoor permutations and claw-free functions

Collection of OWF

✧ Def: Collection of OWF

$\{f_i : D_i \rightarrow \{0,1\}^*\}_{i \in \hat{I}}$ is strongly one-way if

$\exists$ three PPT algorithms (I , D , F), s.t.

1. Easy to sample and compute

   I, on input 1^n, randomly selects an index $i \in \hat{I} \cap \{0,1\}^{p(n)}$
   D, on input $i \in \hat{I}$, randomly selects $x \in D_i$
   F, on input $i \in \hat{I}$ and $x \in D_i$ returns $f_i(x)$ (F may be assumed deterministic)

2. Hard to invert

   For every PPT TM A', every positive polynomial p(·), and all sufficient large n

   $\Pr\{A'(I_n, f_{I_n}(X_n)) \in f_{I_n}^{-1} f_{I_n}(X_n)\} < 1 / p(n)$

   I_n is a r.v. of the output of I on input 1^n,
   X_n is a r.v of the output of D, on input I_n

Example Collection of OWF

- RSA Function:
  - An infinite set of indices: $\hat{I}$ pairs (N, e)

    where $N = P \cdot Q$, P and Q are both (½ $\log_2 N$)-bit primes,
    $\gcd(e, \phi(N))=1$, $\phi(N)=(P-1)\cdot(Q-1)$
  - A corresponding set of finite functions, $\{f_i\}_{i\in\hat{I}}$ $x^e \bmod N$
  - For each $i\in\hat{I}$, the domain of f_i, denoted D_i $D_{N,e}=\{1,\ldots,N\}$
  - An efficient algorithm I, on input 1^n, randomly selects $i\in\hat{I}$ I_{RSA} selects uniformly two primes P and Q s.t. $2^{n-1}\le P<Q<2^n$ and an e relatively prime to $\phi(N)$
  - An efficient algorithm D, on input $i\in\hat{I}$, randomly selects $x\in D_i$ D_{RSA} selects uniformly an element in $D_{N,e}=\{1,\ldots,N\}$
  - An efficient function-evaluation algorithm F F_{RSA} on input ((N,e),x) outputs $RSA_{N,e}(x) = x^e \bmod N$

RSA Function

- RSA collection is a collection of permutations
- Invert $RSA_{N,e}$ $\le_T$ Factor(N)

  (break RSA function)

  the other direction of reduction is a well-known open problem
- The best algorithms known for inverting $RSA_{N,e}$ proceed by (explicitly or implicitly) factoring N

Rabin Function

- Indices: N where $N = P \cdot Q$, P and Q are both (½ $\log_2 N$)-bit primes
- Functions: $Rabin_N(x) = x^2 \bmod N$
- Domains: $D_N = \{1, 2, \ldots N\}$
- Index sampling algorithm (key generation): I_{Rabin} selects uniformly two primes P and Q s.t. $2^{n-1}\le P<Q<2^n$
- Domain sampling algorithm: D_{Rabin} selects uniformly $x\in D_N=\{1,\ldots,N\}$
- Function-evaluation algorithm F_{Rabin} input (N,x) outputs

  $Rabin_N(x) = x^2 \bmod N$

- Rabin function is a 4-to-1 mapping
- Extracting square roots modulo N is computationally equivalent to factoring N

Discrete Logarithms

- Indices: P, G where P is a prime, G is a primitive element in Z_P^*
- Functions: $DLP_{P,G}(x) = G^x \bmod P$
- Domains: $D_{P,G} = Z_P^*=\{1, 2, \ldots P-1\}$
- Index sampling algorithm (key generation): I_{DLP} selects uniformly a prime P s.t. $2^{n-1}\le P<2^n$ and a primitive element G in Z_P^*
- Domain sampling algorithm: D_{DLP} selects uniformly $x\in D_{P,G}=\{1,\ldots,P-1\}$
- Function-evaluation algorithm F_{DLP} input ((P,G),x) outputs

  $DLP_{P,G}(x) = G^x \bmod P$

- Extracting discrete logarithms in a finite field is widely believed to be intractable
- If $P = 2Q+1$, Q is a prime, this is believed to be the hardest for DLP

Sophie-Germain Prime Number

Collection of Trapdoor Permutations

✧ Def: Let $\hat{I} \subseteq \{0,1\}^*$ and $\hat{I}_n = \hat{I} \cap \{0,1\}^n$,

a collection of permutations with indices in $\hat{I}$ is a set $\{f_i: D_i \to D_i\}_{i \in \hat{I}}$ such that each f_i is 1-1 on D_i

Such a collection is called a trapdoor permutation if there exist four probabilistic poly-time algorithms I, D, F, F^{-1} s.t.

1. Index and trapdoor selection: For every $n \in \mathbb{I}$, ... trapdoor
$$\Pr\{ I(1^n) \in \hat{I}_n \times \{0,1\}^* \} > 1-2^{-n}$$
2. Sampling in domain: For every $n \in \mathbb{I}$ and $i \in \hat{I}_n$
   (a) $\Pr\{ D(i) \in D_i \} > 1-2^{-n}$
   (b) uniformly distributed in D_i $\Pr\{ D(i)=x \mid D(i) \in D_i \} = 1/|D_i|$
3. Efficient evaluation: For every $n \in \mathbb{I}$, $i \in \hat{I}_n$ and $x \in D_i$
$$\Pr\{ F(i, x) = f_i(x) \} > 1-2^{-n}$$

Collection of Trapdoor Permutations

4. Hard to invert: let I_n be a r.v. describing the distribution of the first element in the output of $I(1^n)$, and $X_n = D(I_n)$
   ✡ Standard/uniform-complexity version:
   For every PPT A', every positive $p(\cdot)$ and all suff. large n's,
$$\Pr\{ A'(I_n, f_{I_n}(X_n)) = X_n \} < 1/p(n)$$
   ✡ Non-uniform-complexity version:
   For every family of poly-size circuits $\{C_n\}_{n \in \mathbb{I}}$,
   every positive polynomial $p(\cdot)$ and all suff. large n's,
$$\Pr\{ C_n(I_n, f_{I_n}(X_n)) = X_n \} < 1/p(n)$$
5. Inverting with trapdoor:
   For every $n \in \mathbb{I}$, any pair (i, t) in the range of $I(1^n)$ s.t. $i \in \hat{I}_n$ and $x \in D_i$
$$\Pr\{ F^{-1}(t, f_i(x)) = x \} > 1-2^{-n}$$

Claw-Free Function

✧ Def: A pair of functions $(f^0(x), f^1(x))$, both are easy to evaluate, have the same range $f^0(D^0) = f^1(D^1)$, yet infeasible to find a *claw* (x_0, x_1), $x_\sigma \in D^\sigma$, such that $f^0(x_0) = f^1(x_1) \in f^\sigma(D^\sigma)$

✧ Def 2.4.6: Claw-Free Collection

✶ Collection of pairs of functions: infinite set of indices $\hat{I}$,
two finite sets D_i^0 and D_i^1 for each i
set of function pairs f_i^0 and f_i^1 defined over D_i^0, D_i^1

✶ Claw-free collection of pairs of functions: $\exists$ PPT algorithms I, D, F s.t.

a. Easy to sample and compute
r.v. $I(1^n) = I_n$ distributed over $\hat{I} \cap \{0,1\}^n$
r.v. $D(\sigma, i)$ outputs x_σ distributed over D_i^σ, $\sigma \in \{0, 1\}$
r.v. $F(\sigma, I_n, x_\sigma) = f_i^\sigma(x_\sigma)$ for each $x_\sigma \in D_i^\sigma$

b. Identical range distribution
$\forall\, i \in \hat{I} \cap \{0,1\}^n$, $f_i^0(D(0,i))$ and $f_i^1(D(1,i))$ are identically distributed

Claw-Free Collection (cont'd)

c. Hard to form claws
a pair (x_0, x_1), $x_\sigma \in D^\sigma$ satisfying $f_i^0(x_0) = f_i^1(x_1)$ is called a claw for index I, C_i is the set of claws for index $i \in \hat{I}$

$\forall$ PPT A', $\forall\, p(\cdot)$, every sufficiently large n,
$$\Pr\{ A'(I_n) \in C_{I_n} \} < 1/p(n)$$

Note: 1. b, c are conflicting requirements: b says that claws do exists, c says claws can not be efficiently found
2. A claw-free collection of functions yields a collection of strong OWF, Exercise 2-22
3. If $D_i = D_i^0 = D_i^1$, the domains for both function are the same and f_i^0, f_i^1 are permutations over D_i then such a collection is called a collection of claw-free pair of permutations

CFC exists ⇒ OWF exists

- If it is infeasible to find claws for $f_i^0(\cdot)$ and $f_i^1(\cdot)$ then it is infeasible to invert either permutations $f_i^0(\cdot)$ or $f_i^1(\cdot)$

pf:

i.e. an inversion algorithm enables one to create claw easily

assume that $(f_i^0)^{-1}$ is the inversion of f_i^0, you can randomly pick x_1 and calculate $x_0 = (f_i^0)^{-1}(f_i^1(x_1))$, (x_0, x_1) is a claw

- Being claw-free is stronger than being one-way

ex. Two RSA functions $f_i^0(x) \equiv x^s \pmod n$, $f_i^1(x) \equiv x^t \pmod n$ are both hard to invert, but apparently not claw free since the modulo exponentiation is commutative claw: (r^t, r^s)

Claw-Free Perm induces OWTP

- Practically, all known examples of trapdoor permutation are induced by corresponding families of claw-free permutations.
  - Y. Dodis and L. Reyzin, "On the power of claw-free permutations," Security in Communication Networks, 2002
- For example, CFP from RSA function
  - Modulus n, exponent e, gcd(e, ϕ(n)) = 1, $y \in_R Z_n^*$
  - $f(x) \equiv x^e \pmod n$
  - $g(x) \equiv x^e \cdot y \pmod n$

pf.

if we can find x_0 and x_1 satisfy $f(x_0) = g(x_1)$,

then $(x_0/x_1)^e = y \pmod n$

and we find the e-th root of y as x_0/x_1

i.e. if RSA is a trapdoor permutation, then (f, g) is CFP

DLP Claw-Free Collection

- Assuming that the DLP for fields of Sophie-Germain prime cardinality is intractable
- Index set: P is a prime, G is a primitive, Z is a field element in Z_p^*
- Index sampling algorithm: select P, G and select Z uniformly in Z_p^*
- Domain: same for both functions with index (P, G, Z)
- Domain sampling algorithm: uniform
- Function pairs: $f^{\sigma}_{P,G,Z}(x) \equiv Z^{\sigma} \cdot G^x \bmod p$, $\sigma \in \{0, 1\}$

- The ability to form a claw (x_0, x_1) for the index (P, G, Z) yields the ability to find the discrete log of Z
  - $G^{x_0} \equiv Z \cdot G^{x_1} \pmod p \Rightarrow Z \equiv G^{x_0 - x_1} \pmod p$
  - If the DLP is a one-way collection then the above function pairs are claw-free collection

Claw-Free Collections based on Factoring

- Legendre symbol of r modulo a prime P: $LS_P(r) \equiv_P r^{(p-1)/2}$ is 1 if r is a quadratic residue modulo P and is -1 if r is not a quadratic residue modulo P
- Jacobi symbol of r modulo a composite N:

$$JS_N(r) \triangleq \prod_{i=1}^{t} LS_{P_i}(r)^{e_i} \qquad N = \prod_{i=1}^{t} P_i^{e_i}$$

$JS_N(r) = JS_N(r \bmod N)$

$JS_N(a \cdot b) = JS_N(a) \cdot JS_N(b)$, and $JS_N(1) = 1$

- Blum integer N: product of two primes P, Q, both are congruent to 3 modulo 4 (i.e. in the form of 4k+3)
  - Jacobi symbol of -1 mod N equals 1(since -1 is in $QNR_P \cap QNR_Q$)

$(P-1)/2 = 2k+1 \qquad LS_P(-1) \equiv (-1)^{2k+1} \equiv -1 \pmod P$

$(Q-1)/2 = 2k'+1 \qquad LS_Q(-1) \equiv (-1)^{2k'+1} \equiv -1 \pmod Q$

$$JS_N(-1) = LS_P(-1) \cdot LS_Q(-1) = 1$$

C.F.C. based on Factoring (cont'd)

- Half of the square roots of each quadratic residue have their Jacobi symbols being 1 (in $QR_P \cap QR_Q$ or $QNR_P \cap QNR_Q$)

  let a quadratic residue $y \equiv x^2 \pmod N$

  i.e. $y \equiv x^2 \pmod P \equiv x^2 \pmod Q$

  the four square roots of y is $z_1 \equiv x \pmod P \equiv x \pmod Q$

  $z_2 \equiv x \pmod P \equiv -x \pmod Q$

  $z_3 \equiv -x \pmod P \equiv x \pmod Q$

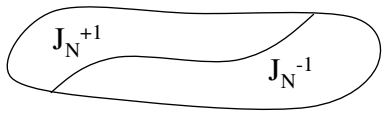
  $z_4 \equiv -x \pmod P \equiv -x \pmod Q$

  Note that $P = 4k + 3$ implies that $JS_P(z) \equiv z^{2k+1} \equiv -JS_P(-z)$

  $JS_N(z_1) = JS_P(x)\, JS_Q(x) = JS_P(-x)\, JS_Q(-x) = JS_N(z_4)$

  $JS_N(z_2) = JS_P(x)\, JS_Q(-x) = JS_P(-x)\, JS_Q(x) = JS_N(z_3) = -JS_N(z_1)$

- J_N^{+1} (J_N^{-1}) denotes the set of elements in Z_N^* with Jacobi symbol +1(-1)



C.F.C. based on Factoring (cont'd)

- Assuming that integer factorization is infeasible
- Example: Claw-Free collections
  - Index set: all Blum integers with two primes of the same length
  - Index sampling algorithm: select P, Q uniformly in $\{0,1\}^n$, $\equiv_4 3$
  - Domain: $J_N^{(-1)^\sigma}$ for $f^\sigma_N(x)$
  - Domain sampling algorithm: uniform in $J_N^{(-1)^\sigma}$
  - Function pairs: $f^\sigma_N(x) \equiv x^2 \bmod N$, $\sigma \in \{0, 1\}$
- The ability to form a claw (x, y) for the index N yields the ability to factor N
  - A claw (x_0, x_1) is formed if $x_0 \in J_N^{+1}$, $x_1 \in J_N^{-1}$ and $x_0^2 \equiv x_1^2 \pmod N$
  - $x_0 \top_N x_1$ since $JS_N(x_0) \neq JS_N(x_1)$
  - $-x_0 \top_N x_1$ since $JS_N(-x_0) = JS_N(-1) \cdot JS_N(x_0) = JS_N(x_0) \neq JS_N(x_1)$

  Therefore, $\gcd(x_1 \pm x_0, N)$ is a nontrivial factor of N

C.F.P. based on Factoring (cont'd)

- Example: A Claw-Free permutation
  - Index set: Blum integer with two primes of the same length, also satisfying $P \equiv_8 3$, $Q \equiv_8 7$ $(\pm 2 \in J_N^{-1})$
  - Index sampling algorithm: select P, Q uniformly in $\{0,1\}^n$, $P \equiv_4 3$, $Q \equiv_8 7$
  - Domain: QR_N for both $f^0_N(x)$ and $f^1_N(x)$
  - Domain sampling algorithm: uniform in QR_N
  - Function pairs: $f^\sigma_N(x) \equiv 4^\sigma \cdot x^2 \bmod N$, $\sigma \in \{0, 1\}$, both are permutations over QR_N
- The ability to form a claw (x_0, x_1) for the index N yields the ability to factor N
  - A claw (x_0, x_1) is formed if $x_0, x_1 \in QR_N$ and $x_0^2 \equiv 4 \cdot x_1^2 \pmod N$
  - $x_0 \top_N 2 \cdot x_1$ since $JS_N(x_0) \neq JS_N(2 \cdot x_1) = JS_N(2) \cdot JS_N(x_1) = (-1) \cdot JS_N(x_0)$
  - $-x_0 \top_N 2 \cdot x_1$ since $JS_N(-x_0) = JS_N(-1) \cdot JS_N(x_0) = JS_N(x_0) \neq JS_N(2 \cdot x_1)$

  Therefore, $\gcd(2 \cdot x_1 \pm x_0, N)$ is a nontrivial factor of N

C.F.P. based on Factoring (cont'd)

- P, Q are large primes in $\{0,1\}^n$, and $P \equiv_4 3$, $Q \equiv_4 3$, then $JS_P(-1) \equiv_P (-1)^{(P-1)/2} \equiv_P (-1)^{2k+1} \equiv_P -1$
- if $x \in QR_N$ then $f^0_N(x) \equiv x^2 \bmod N$ is a permutation

  Assume that you can find $x_1, x_2 \in QR_N$ such that $x_1^2 \equiv_N x_2^2$

  Then,

  $x_1^2 \equiv_P x_2^2$ and $x_1^2 \equiv_Q x_2^2$

  $x_1 \equiv_P \pm x_2$ and $x_1 \equiv_Q \pm x_2$

  i.e.

  $x_1 \equiv_P x_2 \equiv_Q x_2$

  $x_1 \equiv_P x_2 \equiv_Q -x_2$

  $x_1 \equiv_P -x_2 \equiv_Q x_2$

  $x_1 \equiv_P -x_2 \equiv_Q -x_2$

  However, $JS_N(x_1) = JS_N(x_2) = 1$

  and $JS_P(x_1) = JS_Q(x_1) = 1$

  $JS_P(x_2) = JS_Q(x_2) = 1$

  which imply that

  $x_1 \top_P -x_2$ and $x_1 \top_Q -x_2$

  Therefore,

  $x_1 \equiv_P x_2 \equiv_Q x_2$ i.e. $x_1 \equiv_N x_2$

C.F.P. based on Factoring (cont'd)

- P, Q are large primes in $\{0,1\}^n$, $P \equiv_8 3$, $Q \equiv_8 7$, $N=P \cdot Q$
  $JS_N(2) = 1$ if $(N^2-1)/8$ is even and $JS_N(2) = -1$ otherwise
  $16 \nmid (((8k+3)(8k'+7))^2-1)/8$, therefore,
  $JS_N(2) = -1$, i.e. $2 \in QNR_N$
  $JS_N(-2) = JS_N(-1) \cdot JS_N(2) = 1 \cdot (-1) = -1$, i.e. $-2 \in QNR_N$

- if $x \in QR_N$ then $f^1_N(x) \equiv 4\ x^2 \bmod N$ is a permutation
  Assume that you can find $x_1, x_2 \in QR_N$ such that $4\ x_1^2 \equiv_N 4\ x_2^2$
  Then,
  $x_1^2 \equiv_N x_2^2$, since gcd(4, N) = 1
  all the other proof follows
  at last $x_1 \equiv_N x_2$

Secure Signature Scheme based CFP

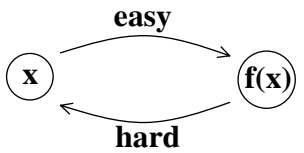
- Extension of a claw-free pair $(f_0(\cdot), f_1(\cdot))$
  $f_i(x) = f_{i_d}(f_{i_{d-1}}(f_{i_{d-2}}(\ldots(f_{i_1}(f_{i_0}(x)))\ldots))), \quad i = i_d\, i_{d-1} \ldots i_1\, i_0$
- Atomic authentication step:
  - Assuming that $(f_0(\cdot), f_1(\cdot))$ are claw-free pair of functions
  - For an arbitrary Q, it is hard to find R, L such that $f_R(L) = Q$ unless you know both f_0^{-1} and f_1^{-1}
  - If someone can provide a set of R, L such that $f_R(L) = Q$ for a specified Q, we believe that he must have the trapdoors of f_0 and f_1
- Simplified signature concept:
  - $\sigma = f_M^{-1}(R)$ is sort of signature for M if R is constrained by some other way, i.e. $f_M(\sigma) = R$
  - let M' and M differ only in LSB, and you have a poly-time algorithm to find a claw (σ, ρ), it satisfies $f_M(\sigma) = R = f_{M'}(\rho)$
  - if R can be chosen by the attacker, (M, σ, R) and (M', ρ, R) are both valid signatures

Summary of CFPs

- RSA: $f(x) \equiv x^e \pmod n$, $g(x) \equiv x^e \cdot y \pmod n$
- Rabin I: $f(x) \equiv x^2 \pmod n$, $g(x) \equiv x^2 \cdot y \pmod n$
- Rabin II: $f(x) \equiv x^2 \pmod n$ for $x \in J_n$, $p \equiv q \equiv 3 \pmod 4$
  $g(x) \equiv x^2 \pmod n$ for $x \in J_n^{-1}$
- Rabin III: $f(x) \equiv x^2 \pmod n$ for $x \in QR_n$, $p \equiv 3, q \equiv 7 \pmod 8$
  $g(x) \equiv x^2 \cdot 4 \pmod n$ for $x \in QR_n$
- Paillier: $f(x) \equiv g^2 \cdot r^n \pmod{n^2}$ for $x \in Z_n^*$,
  $g(x) \equiv g^2 \cdot r^n \cdot y \pmod{n^2}$ for $x \in Z_n^*$
- Discrete logarithm: $f(x) \equiv g^x \pmod p$ for $x \in Z_p^*$,
  $g(x) \equiv g^x \cdot y \pmod p$ for $x \in Z_p^*$

Hardcore Predicate

OWF:



- idea: given f(x), predict a certain bit (or some derived value, b(x)) of x might be easier than predict x directly
- predicate: $b: \{0,1\}^* \rightarrow \{0,1\}$ b(x)
- Hardcore: poly-time computable predicate b is hardcore of a function f(·) if
  for all PPT A', for all p(·), for all sufficiently large n
  $\Pr\{A'(f(U_n)) = b(U_n)\} < 1/2 + 1/p(n)$

- properties of a hardcore predicate b(·): unbiased
  $\Pr\{b(U_n)=0\} = \Pr\{b(U_n)=1\} = 1/2$

OWF exists ⇒ Hardcore Predicate exists

- Consider a special construction of a hardcore predicate function from a OWF f(·)
  - $U(x, r) = x \cdot r \mod 2$ (the XOR of a random subset of x's bits)
    - $z = U(x, r) = (x_1 x_2 \cdots x_n) \cdot (r_1 r_2 \cdots r_n) \bmod 2$
      $= \sum_{i=1}^{n} x_i \cdot r_i \bmod 2 = \bigoplus_{i=1}^{n} x_i \cdot r_i$
    - where $|x| = |r| = n$, r is a random string, $r \in_R \{0,1\}^n$
    - Given f(x), and a random r, is guessing U(x, r) hard? Note: for some fixed r, this problem might be easy, but how about in the average case for random r?
    - Proving plan: by contradiction, assume guessing U(x, r) easy, i.e. there exists an algorithm G, given f(x) and r as inputs, can predict U(x, r) with noticeable prob. We want to use G to construct another algorithm A that can invert f(·) with noticeable probability
    - Therefore, OWF exists ⇒ Hardcore predicate exists

∃ OWF ⇒ ∃ Hardcore Predicate (2/8)

- Properties of U(x, r)
  - $U(x, r) \oplus U(x, r^i) = x_i$
  - $U(x, r_1) \oplus U(x, r_2) = U(x, r_1 \oplus r_2)$

r^i is the same as vector r except the i-th bit, $r^i = (r_1, r_2, \ldots \bar{r}_i, r_{i+1}, \ldots r_n)$

Bitwise XOR of two n-bit strings

$$(x \cdot r_1 \bmod 2 + x \cdot r_2 \bmod 2) \bmod 2$$
$$= (x \cdot r_1 + x \cdot r_2) \bmod 2$$
$$= x \cdot ((r_1 + r_2) \bmod 2) \bmod 2$$

- Pairwise-Independent Sampling: Let $X_1, X_2, \ldots X_n$ be pairwise-independent random variables with the same expectation, denoted μ, and the same variance σ^2. Then for every $\varepsilon > 0$,

$$\Pr\left\{ \left| \frac{\sum_{i=1}^{n} X_i}{n} - \mu \right| \geq \varepsilon \right\} \leq \frac{\sigma^2}{\varepsilon^2 n}$$ Chebyshev Ineq.

Pairwise indep: $\Pr\{x_i = a \wedge x_j = b\} = \Pr\{x_i = a\} \cdot \Pr\{x_j = b\}$

∃ OWF ⇒ ∃ Hardcore Predicate (3/8)

- We will assume XOR of a random subset r is not hardcore, then there exists a G to predict the XOR w.r.t the random r, we need to run this G for polynomial times with pairwisely independent r's in algorithm A for inverting f(·)
  - Randomly select $r_1, r_2, \ldots r_\ell \in \{0,1\}^n$
    where $\ell = \lceil \log_2(2 \cdot n \cdot p(n)^2 + 1) \rceil$
  - Let $J \subseteq \{1, 2, \ldots \ell\}$ and $J \neq \phi$, let $r_J = \bigoplus_{j \in J} r_j$ **n-bit, bitwise XOR**
  - These $2 \cdot n \cdot p(n)^2$ r_J's are pairwisely independent (there is at least one distinct element in J and J', $J \neq J'$ and J, $J' \subseteq \{1, 2, \ldots \ell\}$) and could be used in algorithm A for inverting f(·)

Independence of X and X ⊕ Y (4/8)

- If X, Y are independent binary random variables, will X and X⊕Y be independent?
  - No, unless Y is uniformly distributed, i.e. $\Pr\{Y = \pm 1\} = 1/2$

X	Pr{X}
0	1-p
1	p

Y	Pr{Y}
0	1-q
1	q

X	Y	X⊕Y	Pr{X,Y}
0	0	0	(1-p)(1-q)
0	1	1	(1-p)q
1	0	1	p(1-q)
1	1	0	pq

X⊕Y	Pr{X⊕Y}
0	1-p-q+2pq
1	p+q-2pq

X	X⊕Y	Pr{X,X⊕Y}	Pr{X}·Pr{X⊕Y}
0	0	(1-p)(1-q)	(1-p)(1-p-q+2pq)
0	1	(1-p)q	(1-p)(p+q-2pq)
1	0	pq	p(1-p-q+2pq)
1	1	p(1-q)	p(p+q-2pq)

p=1/2	
LHS	RHS
(1-q)/2	1/4
q/2	1/4
q/2	1/4
(1-q)/2	1/4

q=1/2	
LHS	RHS
(1-p)/2	(1-p)/2
(1-p)/2	(1-p)/2
p/2	p/2
p/2	p/2

$\Pr\{X, X \oplus Y\} \neq \Pr\{X\} \cdot \Pr\{X \oplus Y\}$ except q=1/2

The Inverting Algorithm A (5/8)

- ✧ Given y=f(x), let G be an efficient algorithm to guess U(x, r)
- ✧ Construct A(y) let $n=|y|$ and $\ell = \lceil \log_2(2\cdot n\cdot p(n)^2+1) \rceil$
  - ✶ Randomly generate $r_1, r_2, \ldots r_\ell \in \{0,1\}^n$,
  - ✶ Guess $\sigma_j = U(x, r_j)$, $j=1,2,\ldots \ell$ $\Pr\{\sigma_j \text{ all correct}\} = 2^{-\ell}$
  - ✶ Form other $2^\ell - \ell - 1$ $r_J = \bigoplus_{j\in J} r_j$ where $J\subseteq\{1, 2, \ldots \ell\}$ and $J\neq\phi$
  - ✶ Also calculate $\sigma_J = U(x, r_J) = \bigoplus_{j\in J} \sigma_j$ from property two of $U(\cdot,\cdot)$
    if σ_j's are correct, then σ_J's are correct, $\Pr\{\sigma_J \text{ all correct}\} = 2^{-\ell}$
    now we have $R = \{r_{J_1}, r_{J_2}, \ldots r_{J_m}\}$ $m=2\cdot n\cdot p(n)^2$ elements $=2^\ell-1$
  - ✶ For i = 1, 2, …n
    for all $r_J \in R$, calculate the J-th estimate of bit i of x as
    $$z_{iJ} = G(y=f(x), r_J^i) \oplus \sigma_J$$
    Take majority for each bit, i.e. $\hat{x}_i = 1$ if $\sum_{J\subseteq\{1,\ldots\ell\}} z_{iJ} / (2\cdot n\cdot p(n)^2) > 1/2$

Good Set for G(f(x), r) (6/8)

- ✧ Let $x \in \{0,1\}^n$
- ✧ Define $s_n = \{x: \Pr\{G(f(x), r) = U(x, r)\} \geq \frac{1}{2} + \frac{1}{2p(n)}\}$ (over all possible $r\in\{0,1\}^n$)
- ✧ $|s_n| = ?$
  - ✶ If $x \in s_n$, $\Pr\{G \text{ is correct}\} \geq \frac{1}{2} + \frac{1}{2p(n)}$
  - ✶ If $x \in \{0,1\}^n - s_n$, $\Pr\{G \text{ is correct}\} < \frac{1}{2} + \frac{1}{2p(n)}$

$\Rightarrow \forall\, x\in\{0,1\}^n$, $\Pr\{G \text{ is correct}\} < (|s_n|\cdot 1 + (2^n-|s_n|)\cdot(\frac{1}{2} + \frac{1}{2p(n)}))/2^n$

Assume $U(\cdot,\cdot)$ is not a hardcore, i.e. $\frac{1}{2} + \frac{1}{p(n)} \leq \Pr\{G \text{ is correct}\}$

$\Rightarrow 2^n (\frac{1}{2} + \frac{1}{p(n)}) < (|s_n| + (2^n-|s_n|)(\frac{1}{2} + \frac{1}{2p(n)}))$

$\Rightarrow \cancel{\frac{2^n}{2}} + \frac{2^n}{p(n)} < |s_n|/2 - \frac{|s_n|}{2p(n)} + \cancel{\frac{2^n}{2}} + \frac{2^n}{2p(n)}$

$\Rightarrow \frac{2^n}{2p(n)} < |s_n| \cdot (\frac{1}{2} - \frac{1}{2p(n)})$

$\Rightarrow |s_n| > 2^n / \left(p(n)\cdot(1-\frac{1}{p(n)})\right) = 2^n / (p(n)-1) > 2^n / p(n)$

Note: The probability $|s_n| / 2^n$ is noticeable

Algorithm A Inverts f(·) Noticeably (7/8)

- ➢ Assume $U(\cdot,\cdot)$ is not a hardcore then $\frac{1}{2} + \frac{1}{p(n)} \leq \Pr\{G \text{ is correct}\}$
  $2\cdot n\cdot p(n)^2$ and $|s_n| > 2^n / p(n)$
- ➢ For an $x \in s_n$, perform polynomial times of G for pairwisely indep. r_j's to estimate the i-th bit of x (use the ⊕ of bits in the power set R as r_J)
  $U(x, r_J) \oplus G(f(x), r_J^i) = z_{iJ}$ Define the event $\xi_J=1$ if $z_{iJ} = x_i$
- ➢ Algorithm A uses the majority rule to decide $\hat{x}_i$, the probability that this output is incorrect,
  $$E_i = \Pr\{\sum_J \xi_J \leq m/2\} \leq \Pr\{|\sum_J \xi_J - (\frac{1}{2} + \frac{1}{2p(n)} + \varepsilon)\, m| \geq \frac{m}{2p(n)}\}$$
  $$= \Pr\left\{\left|\frac{\sum_J \xi_J}{2^\ell-1} - \left(\frac{1}{2} + \frac{1}{2p(n)} + \varepsilon\right)\right| \geq \frac{1}{2p(n)}\right\} \leq \frac{\mathrm{Var}(\xi_J)}{\left(\frac{1}{2p(n)}\right)^2\cdot(2^\ell-1)} \leq \frac{1}{2n}$$
  $2^\ell - 1$; $\mathrm{Var}(\xi_J) \leq \frac{1}{4}$; $2^\ell = 2\cdot n\cdot p(n)^2 + 1$
- ➢ $\Pr\{\text{at least one bit has error} \mid x \in s_n, \{\sigma_j\} \text{ correct}\} =$
  $\Pr\{E_1 \cup E_2 \cup \ldots \cup E_n\} \leq \sum_{i=1}^n \Pr\{E_i\} \leq n \cdot \frac{1}{2n} = \frac{1}{2}$
- ➢ $\Pr\{A \text{ is correct}\} > \frac{1}{p(n)} \cdot 2^{-\ell} \cdot (1-\frac{1}{2}) = \frac{1}{q(n)}$ ¶
  $x\in s_n$ $\{\sigma_i\}$ correct all $\hat{x}_i$ correct

∃ OWF ⇒ ∃ Hardcore Predicate (8/8)

- ✧ Therefore, we have the following theorem proved

- ✧ Thm: If f(x) is a strong one-way function, h(x, r)=(f(x), r)
  is still a strong one-way function, $U(x, r) = \bigoplus_{i=1}^n x_i \cdot r_i$
  is a hardcore predicate for $h(\cdot, \cdot)$

  - ➢ It is infeasible to guess the XOR of a random subset of the bits of x when given f(x) and the random subset specified by r
  - ➢ In the proof, the key idea is that we only guess ℓ bits σ_i the success rate is $\frac{1}{2\cdot n\cdot p(n)^2+1}$, which is a noticeable figure, instead of guessing n bits, the success rate is only 2^{-n}

Historical Notes

- Diffie & Hellman: notion of OWF, trapdoor permutation
- Yao: weak OWF, equivalence of weak and strong OWF, OWF⇒hardcore predicate
- Rivest, Shamir, Adelman: RSA function
- Rabin: primality test, randomized algorithm, Rabin function
- Blum: hardcore predicate, Math
- Micali: hardcore predicate, formalization of cryptography
- Goldwasser: ZKP system, formalization of cryptography
- Levin: Math
- Goldreich & Levin: XOR of random subset is hardcore

Yao, Rabin, and Blum had been awarded the Turing Award

Bit Security

- Discrete Log
  - Blum and Micali, How to Generate Sequences Of Cryptographically Strong Pseudo-Random Bits, SIAM J. on Computing, Vol. 13, Nov. 1984, pp.850-864.
  - Peralta, Simultaneous Security of Bits in the Discrete Log, Eurocrypt'85
  - Boneh, Hardness of Computing the Most Significant Bits of Secret Keys in Diffie-Hellman and Related Schemes, Crypto'96
- RSA
  - Ben-Or, On the cryptographic security of single RSA bits, STOC'83
  - Chor and Goldreich, RSA/Rabin Bits Are 1/2 + 1/poly(log N) Secure, SIAM JC ??, FOCS84, pp.449-463
  - Chor, RSA/Rabin least significant bits are ½+1/poly(logN) secure, Crypto'84
  - Alexi, RSA and Rabin Functions: Certain parts are as hard as the whole, SIAM J. Computing 1988
  - Hastad, The Security of all RSA and Discrete Log Bits, FOCS'98
  - Hastad, The Security of Individual RSA Bits, FOCS'98
- Naslund, All Bits in ax+b mod p are Hard, Crypto'96