up to a negligible fraction of inputs

Now, a digression... appealing property of discrete log problem: either it is <u>hard everywhere</u> or <u>hard nowhere</u>.

- Suppose we have an efficient algorithm A

$$\Pr[x \xleftarrow{R} \mathbb{Z}_p : A(g, g^x) \to x] = \varepsilon \quad (\text{for non-negligible } \varepsilon)$$

- We can use A to build B that solves <u>any</u> discrete log instance arbitrarily close to 1:
  - On input $(g, h = g^x)$, B samples $y \xleftarrow{R} \mathbb{Z}_p$ and runs A on (g, h^y)
  - Since y is uniform, g^y is a uniform group element so

$$\Pr[A(g, h^y) \to xy] = \varepsilon$$

  If A succeeds (e.g., outputs $t = xy$ where $h^y = g^t$, then A outputs $x = y^{-1}t$
  - A can repeat this process n/ε times so the success probability becomes

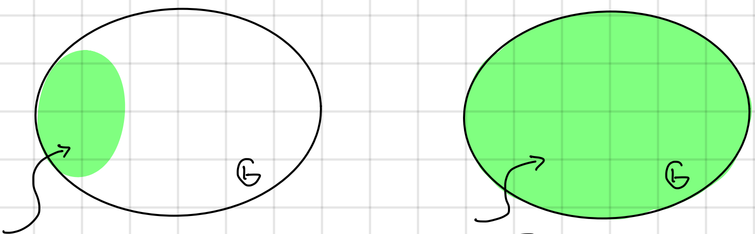
$$1 - (1-\varepsilon)^{n/\varepsilon} \leq 1 - e^{-n} \qquad [\text{since } 1 + x \leq e^x \text{ for all } x]$$

- <u>Conclusion</u>: discrete log either easy everywhere or hard everywhere
  ↳ easy on non-negligible fraction ⇒ easy everywhere

"random self-reduction:" reduce problem to <u>random</u> instance of itself

<u>Implication</u>: instead of assuming that most instances are hard, it suffices that at least an ε-fraction of instances are hard for any non-negligible ε

<u>Visually</u>:



Algorithm A works on an ε-fraction of $\mathbb{G}$

Algorithm B work <u>everywhere</u> in $\mathbb{G}$

In cryptography, we need problems that are hard in the <u>average</u> case (nearly all keys are "good")
↳ differs from worst-case hardness (e.g., NP-hardness — many NP-complete problems believed to be hard in worst case, but good algorithms exist for "typical" instances — <u>not</u> a good basis for crypto)
↳ when a problem has a random self-reduction, then worst-case hardness effectively implies average-case hardness; cannot have setting where problem is easy on ε-fraction of instances for any non-negligible ε)
  ↳ appealing property for crypto!

This is an example of a "random self-reduction" — we can generate random instances of the problem from <u>any</u> instance
↳ this is often times a <u>very</u> useful property

An algebraic PRF with <u>many</u> useful properties:
- Let $\mathbb{G}$ be a group of prime order p and generator g
- PRF key is a random exponent $k \xleftarrow{R} \mathbb{Z}_p$
- $\mathrm{PRF}(k, x) := H(x)^k$ where $H : \{0,1\}^n \to \mathbb{G}$ is a hash function (modeled as a random oracle)

$\forall x : H(x)$ is uniformly distributed over $\mathbb{G}$

Before proving its security, let's look at some useful properties of this PRF:

- Supports oblivious evaluation:

$$\begin{array}{ccc} \underline{\text{client } (x)} & & \underline{\text{server } (k)} \\ r \xleftarrow{R} \mathbb{Z}_p & & \\ & \xrightarrow{\quad H(x)^r \quad} & \leftarrow \text{uniform random group element (independent of } x) \\ & \xleftarrow{\quad H(x)^{rk} \quad} & \end{array}$$

$$\left[H(x)^{rk} \right]^{r^{-1}} = H(x)^k = \mathrm{PRF}(k, x)$$

Useful for anonymous credentials

- Cloudflare's Privacy Pass (browser plugin): solve CAPTCHA $\rightarrow$ obtain set of authentication tokens $(x_1, \mathrm{PRF}(k, x_1)), \ldots (x_n, \mathrm{PRF}(k, x_n))$
  - ↳ tokens cannot be linked to user

- Key-homomorphic:

$$\left.\begin{array}{l} \mathrm{PRF}(k_1, x) = H(x)^{k_1} \\ \mathrm{PRF}(k_2, x) = H(x)^{k_2} \end{array}\right\} H(x)^{k_1} \cdot H(x)^{k_2} = H(x)^{k_1 + k_2} = \mathrm{PRF}(k_1 + k_2, x)$$

Useful building block for updatable encryption:

- Server has many ciphertexts encrypted under k_1

$$\begin{array}{lcl} ct_1 = (x_1, \mathrm{PRF}(k, x_1) \cdot m_1) & & ct_1' = (x_1, \mathrm{PRF}(k', x_1) \cdot m_1) \\ \vdots \qquad\qquad \vdots & \Longrightarrow & \vdots \qquad\qquad \vdots \\ ct_n = (x_n, \mathrm{PRF}(k, x_n) \cdot m_n) & \uparrow & ct_n' = (x_n, \mathrm{PRF}(k', x_n) \cdot m_n) \end{array}$$

client sends $k' - k$

↳ server can non-interactively perform a key-rotation (without decrypting ciphertexts!)

The random oracle model: assume parties have oracle access to a truly random function

↳ can view oracle as maintaining a truth table that is lazily sampled (namely, if $x \in \{0,1\}^n$ is in the table T, reply with $T(x)$; otherwise sample $y \xleftarrow{R} \mathbb{G}$, add $(x \mapsto y)$ to T and reply with y)

adversaries are modeled as oracle Turing machines (with a randomness tape)

↳ success probability taken over its own randomness and the randomness of oracle outputs

when a reduction runs an adversary, it is responsible for answering oracle queries by the adversary

Theorem. If the DDH assumption holds in $\mathbb{G}$, and H is modeled as a random oracle, then $PRF(k, x) := H(x)^k$ is a secure PRF.

Proof Idea. DDH assumption: (g, g^a, g^b, g^{ab}) indistinguishable from (g, g^a, g^b, g^r) where $a, b, r \xleftarrow{R} \mathbb{Z}_p$

$\hookrightarrow$ try to set $k \mapsto a$, $H(x) \mapsto g^b$ $\implies$ then $H(x)^k = g^{ab}$

$\hookrightarrow$ which is indistinguishable from uniform under DDH

Problem: PRF adversary can make multiple queries and need to answer all of them using a single DDH challenge

Approach: Use a random self reduction to randomize (g^b, g^{ab}): [a is fixed in PRF]

Given (g, h, u, v) where $h = g^a$, $u = g^b$, and $v = g^{ab}$ or $v = g^r$:

Sample $s, t \xleftarrow{R} \mathbb{Z}_p$ and output $(g, h, u^s g^t, v^s h^t)$.

Case 1: $v = g^{ab}$ so $(g, h, u^s g^t, v^s h^t)$

$$= (g, g^a, g^{bs+t}, g^{abs+at})$$

$$= (g, g^a, g^{bs+t}, g^{a(bs+t)})$$

$\uparrow$ t is uniform in $\mathbb{Z}_p$ so this is a fresh DDH tuple

Case 2: $v = g^r$ so $(g, h, u^s g^t, v^s h^t)$

$$= (g, h, g^{bs+t}, g^{rs+at})$$

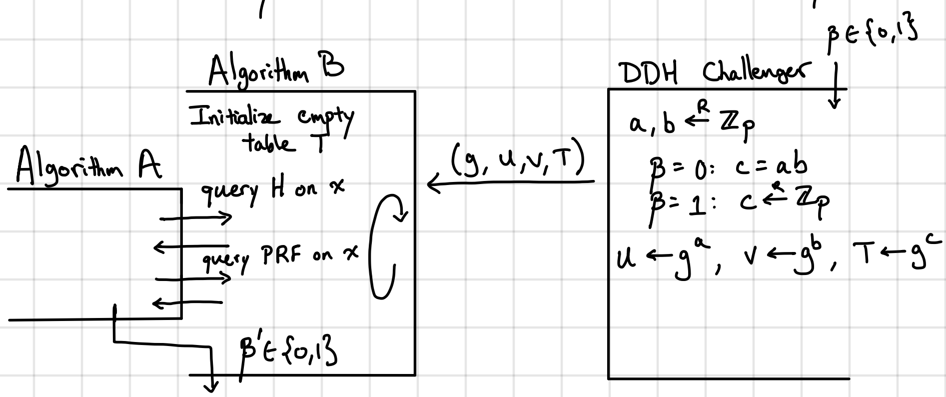
uniform over $\mathbb{G}$ since t is uniform

independent and uniform over $\mathbb{Z}_p$ whenever $r \neq ab$

$$\begin{bmatrix} b & 1 \\ r & a \end{bmatrix} \begin{bmatrix} s \\ t \end{bmatrix}$$

linearly independent (invertible) if $r \neq ab$

Proof. Let A be a PRF adversary. We use A to construct an adversary B for DDH:



When A queries H on x:

1. Check if $x \mapsto (y, z)$ is in the table. If so, reply with y.
2. Otherwise, sample $s, t \xleftarrow{R} \mathbb{Z}_p$ and add $x \mapsto (u^s g^t, v^s h^t)$ to the table.
3. Reply to A with $u^s g^t$.

When A queries PRF on x:

1. Check if $x \mapsto (y, z)$ is in the table. If so, reply with z.
2. Otherwise, sample $s, t \xleftarrow{R} \mathbb{Z}_p$ and add $x \mapsto (u^s g^t, v^s h^t)$ to the table.
3. Reply to A with $v^s g^t$

By the analysis above, if $T = g^{ab}$, then B perfectly simulates the PRF security game where the key is a and $H(x)$ is $u^s g^t$ for $s, t \xleftarrow{R} \mathbb{Z}_p$ (namely, $H(x)$ is random group element). If $T \xleftarrow{R} \mathbb{G}$, then the responses to the PRF queries are uniform and independent of x (from the analysis of the self-reduction above).