CS 6501 Week 3: Number-Theoretic Cryptography

<u>So far in the course</u>: we have mechanisms for message confidentiality and integrity, but all rely on parties having a shared key

<u>Question</u>: Where do symmetric keys come from?

We will begin with a few concepts from algebra that will be useful:

<u>Definition</u>. A group consists of a set $\mathbb{G}$ together with an operation $*$ that satisfies the following properties:
- <u>Closure</u>: If $g_1, g_2 \in \mathbb{G}$, then $g_1 * g_2 \in \mathbb{G}$
- <u>Associativity</u>: For all $g_1, g_2, g_3 \in \mathbb{G}$, $g_1 * (g_2 * g_3) = (g_1 * g_2) * g_3$
- <u>Identity</u>: There exists an element $e \in \mathbb{G}$ such that $e * g = g = g * e$ for all $g \in \mathbb{G}$
- <u>Inverse</u>: For every element $g \in \mathbb{G}$, there exists an element $g^{-1} \in \mathbb{G}$ such that $g * g^{-1} = e = g^{-1} * g$

In addition, we say a group is commutative (or abelian) if the following property also holds:
- <u>Commutative</u>: For all $g_1, g_2 \in \mathbb{G}$, $g_1 * g_2 = g_2 * g_1$

called "multiplicative" notation

<u>Notation</u>: Typically, we will use "$\cdot$" to denote the group operation (unless explictly specified otherwise). We will write g^x to denote $\underbrace{g \cdot g \cdot g \cdots g}_{x \text{ times}}$ (the usual exponential notation). We use "1" to denote the <u>multiplicative identity</u>.

<u>Examples of groups</u>: $(\mathbb{R}, +)$: real numbers under addition
$(\mathbb{Z}, +)$: integers under addition
$(\mathbb{Z}_p, +)$: integers modulo p under addition [sometimes written as $\mathbb{Z}/p\mathbb{Z}$)

here, p is prime

<u>The structure of $\mathbb{Z}_p^*$</u> (an important group for cryptography):

$\mathbb{Z}_p^* = \{x \in \mathbb{Z}_p : \text{there exists } y \in \mathbb{Z}_p \text{ where } xy = 1 \pmod{p}\}$

the set of elements with multiplicative inverses modulo p

a, b can be computed <u>efficiently</u> using Euclid's algorithm

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

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

<u>Proof</u>. Take any $x \in \{1, 2, \ldots, p-1\}$. 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 $ax = 1 \pmod{p}$ so $a = x^{-1} \pmod{p}$.

cyclic groups are commutative

defined to be the identity element

<u>Definition</u>. A group $\mathbb{G}$ is <u>cyclic</u> if there exists a <u>generator</u> g such that $\mathbb{G} = \{g^0, g^1, \ldots, g^{|\mathbb{G}|-1}\}$.

<u>Definition</u>. For an element $g \in \mathbb{G}$, we write $\langle g \rangle = \{g^0, g^1, \ldots, g^{|\mathbb{G}|-1}\}$ to denote the set generated by g (which need not be the entire set. The cardinality of $\langle g \rangle$ is the <u>order</u> of g (i.e., the size of the "subgroup" generated by g)

↳ means that $g^{\text{ord}(g)} = 1$

<u>Example</u>. Consider $\mathbb{Z}_7^* = \{1, 2, 3, 4, 5, 6\}$. In this case,

$\langle 2 \rangle = \{1, 2, 4\}$ [2 is <u>not</u> a generator of $\mathbb{Z}_7^*$] $\text{ord}(2) = 3$

$\langle 3 \rangle = \{1, 3, 2, 6, 4, 5\}$ [3 is a generator of $\mathbb{Z}_7^*$] $\text{ord}(3) = 6$

<u>Lagrange's Theorem</u>. For a group $\mathbb{G}$, and any element $g \in \mathbb{G}$, $\text{ord}(g) \mid |\mathbb{G}|$ (the order of g is a divisor of $|\mathbb{G}|$).

↳ For $\mathbb{Z}_p^*$, this means that $\text{ord}(g) \mid p-1$ for all $g \in \mathbb{G}$

The discrete log problem. Let $\mathbb{G}$ be a group and take elements $g, h \in \mathbb{G}$. The discrete log problem in $\mathbb{G}$ is to compute $x \in \mathbb{Z}_{\mathrm{ord}(\mathbb{G})}$ such that $h = g^x$.

GroupGen samples a description of the group

The discrete log assumption in $\mathbb{Z}_p^*$. Sample $(g, p) \leftarrow \mathrm{GroupGen}(1^\lambda)$, where $\log p = \mathrm{poly}(\lambda)$ and $\langle g \rangle = \mathbb{Z}_p^*$. Then, for all efficient adversaries $\mathcal{A}$,

$$\Pr[h \xleftarrow{R} \mathbb{Z}_p^* ;\ x \leftarrow \mathcal{A}(p, g, h) : h = g^x] = \mathrm{negl}(\lambda).$$

Common setting: choose p to be a "safe prime" ($p = 2q+1$, where q is also prime)

- ↳ Avoid: when $p-1$ is "smooth" (splits into product of small primes), there are efficient algorithms for discrete log
- ↳ At 128-bits of security, p is usually ~ 3072 bits (much longer keys → will motivate elliptic-curve crypto)

↳ In fact, more common to work with prime-order groups (e.g., a subgroup of prime order q in $\mathbb{Z}_p^*$ when $p = 2q+1$)

Diffie-Hellman Key Exchange: Let $\mathbb{G}$ be a group of prime order p with generator g:

Alice		Bob
$x \xleftarrow{R} \mathbb{Z}_p$		$y \xleftarrow{R} \mathbb{Z}_p$
	$\xrightarrow{\quad g^x \quad}$	
	$\xleftarrow{\quad g^y \quad}$	
↓		↓
derive a key from g, g^x, g^y, g^{xy}		derive a key from g, g^x, g^y, g^{xy}

[defer key-derivation details for now]

Claim: An eavesdropper who sees g, g^x, g^y (but does not know x or y) cannot derive the shared key (in particular, eavesdropper should not be able to compute g^{xy}).

Observe: Security of protocol requires hardness of discrete log in $\mathbb{G}$ (why?). However, discrete log by itself may not be sufficient. We require that g^{xy} is hard to compute given g, g^x, g^y → this is the "Computational Diffie-Hellman" (CDH) problem

outputs group description (including order p and generator g)

Computational Diffie-Hellman (CDH) assumption: Let $(\mathbb{G}, g, p) \leftarrow \mathrm{GroupGen}(1^\lambda)$. Then, the CDH assumption holds in $\mathbb{G}$ if for all efficient adversaries $\mathcal{A}$,

$$\Pr[x, y \xleftarrow{R} \mathbb{Z}_p ;\ h \leftarrow \mathcal{A}((\mathbb{G}, g, p), g^x, g^y) : h = g^{xy}] = \mathrm{negl}(\lambda)$$

CDH assumption in a group $\mathbb{G}$ says given g, g^x, g^y, hard to compute g^{xy}.

How do we construct a key-derivation function? Typically use a hash function $H : \{0,1\}^* \to \{0,1\}^\lambda$

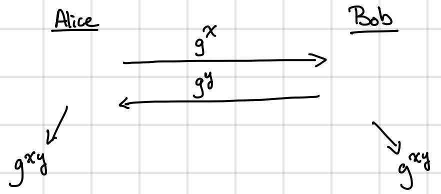
↳ For instance, shared key is $k \leftarrow H(g, g^x, g^y, g^{xy})$.

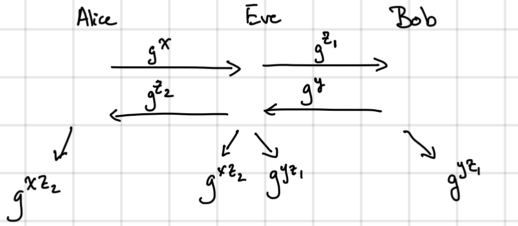
To argue security of Diffie-Hellman key-exchange protocol, we need to assume something about H:

- Option 1: Make the Hash-DH assumption: given g, g^x, g^y, $H(g, g^x, g^y, g^{xy})$ is indistinguishable from random
- Option 2: Model H as a "random oracle" (an ideal object that implements a truly random function)
  - ↳ In this model, if adversary cannot query H on (g, g^x, g^y, g^{xy}), then $H(g, g^x, g^y, g^{xy})$ is uniformly random and completely hidden from the view of the adversary.
  - ↳ Security of DH key-exchange thus follows from CDH assumption in the random oracle model

Diffie-Hellman key-exchange is an anonymous key-exchange protocol: neither side knows <u>who</u> they are talking to

↳ vulnerable to a "man-in-the-middle" attack





Observe Eve can now decrypt all of the messages between Alice and Bob and Alice + Bob have <u>no</u> idea!

What we require: <u>authenticated</u> key-exchange (not anonymous) and relies on a root of trust (e.g., a certificate authority)

↳ On the web, one of the parties will <u>authenticate</u> themself by presenting a <u>certificate</u>

↳ Discussed in greater detail in computer security / applied crypto course (ask in OH if this is interesting)

<u>Public-key encryption</u>: In symmetric encryption, only holder of secret key can encrypt. In public-key encryption, <u>everyone</u> can encrypt, and secret key is only needed for decryption. [Example application: encrypted email]

<u>Definition.</u> A public-key encryption (PKE) scheme consists of three algorithms (KeyGen, Encrypt, Decrypt) with the following properties:

- $\text{KeyGen}(1^\lambda) \to (pk, sk)$: Generates a public key pk and a secret key sk.
- $\text{Encrypt}(pk, m) \to ct$: Takes the public key pk and a message m and outputs a ciphertext ct.
- $\text{Decrypt}(sk, ct) \to m$: Takes the secret key and a ciphertext ct and outputs a message m.

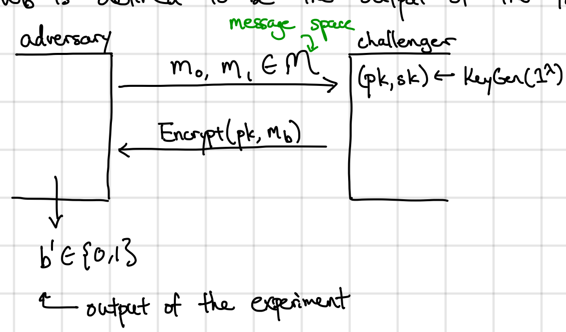
We say the PKE scheme is <u>correct</u> if for all messages m,

$$\Pr[(pk, sk) \leftarrow \text{Setup}(1^\lambda) : \text{Decrypt}(sk, \text{Encrypt}(pk, m)) = m] = 1.$$

We say that the scheme is <u>semantically secure</u> if for all efficient adversaries $\mathcal{A}$,

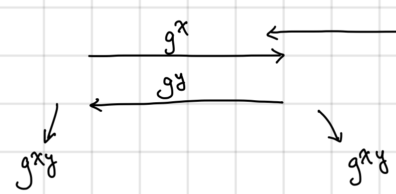
$$\text{PKEAdv}[\mathcal{A}] = |W_0 - W_1| = \text{negl}(\lambda)$$

where W_b is defined to be the output of the following experiment:



<u>Observations.</u>
- For public-key encryption, semantic security implies CPA-security. [Follows via a hybrid argument – check this!]
- Semantically secure PKE schemes <u>must</u> be <u>randomized</u>. [Check this!]

<u>PKE from Diffie-Hellman (ElGamal Encryption)</u>:



<u>Observation</u>: What if we reuse the same g^x for multiple sessions ("static" Diffie-Hellman)?

<u>Idea</u>: Let g^x be the public key and use g^{xy} to hide the message.

ElGamal Encryption. Let $\mathbb{G}$ be a group of prime order p. We construct a PKE scheme as follows:

KeyGen(1^λ): Sample $x \leftarrow \mathbb{Z}_p$ and set $h = g^x$. [1st DH key-exchange message]

Output $pk = h$ and $sk = x$.

Encrypt(pk, m): Choose $y \leftarrow \mathbb{Z}_p$. Output $ct = (g^y, H(g, h, g^y, h^y) \oplus m)$ [2nd DH key-exchange message]

assume $H : \mathbb{G} \to \{0,1\}^n$ and $m \in \{0,1\}^n$

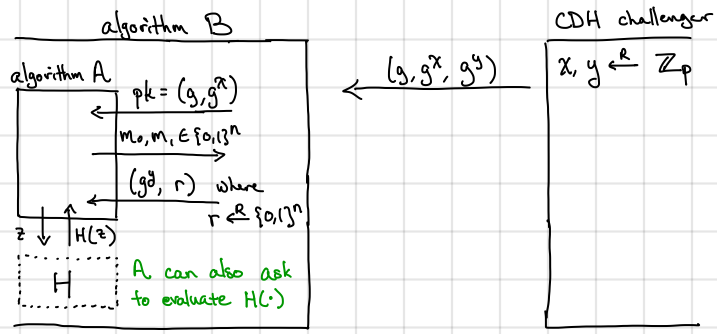
Decrypt(sk, ct). Write $ct = (ct_0, ct_1)$ and compute $ct_1 \oplus H(g, h, ct_0, ct_0^x)$

Correctness: Take any message $m \in \{0,1\}^n$ and $(pk, sk) \leftarrow$ KeyGen(1^λ). If we compute $ct \leftarrow$ Encrypt(pk, m), we have $ct = (g^y, H(g, g^x, g^y, g^{xy}) \oplus m)$. The decryption algorithm then computes

$$\left[H(g, g^x, g^y, g^{xy}) \oplus m\right] \oplus H(g, g^x, g^y, g^{yx}) = m$$

Security. Follows from CDH in the random oracle model.

Proof (Sketch). Suppose we have adversary A that breaks semantic security. We use A to construct an adversary B that breaks CDH in $\mathbb{G}$:



In the random oracle model, if A does not query $H(z)$ for any z, then value of $H(z)$ is uniformly random to A. Thus, message is hidden information-theoretically unless A queries $H(\cdot)$ at (g, g^x, g^y, g^{xy}). In this case, B learns g^{xy} and succeeds in answering the CDH challenge.

↳ Proof shows that the random oracle can be used to extract information from an adversary.

Security without random oracles? Make a stronger assumption.

Decisional Diffie-Hellman: Let $(\mathbb{G}, p, g) \leftarrow$ GroupGen(1^λ). Then, the decisional Diffie-Hellman (DDH) assumption holds in $\mathbb{G}$ if for all efficient adversaries A:

$$\{x, y \xleftarrow{R} \mathbb{Z}_p : (g, g^x, g^y, g^{xy})\} \overset{c}{\approx} \{x, y, z \xleftarrow{R} \mathbb{Z}_p : (g, g^x, g^y, g^z)\}$$

Namely, not only if g^{xy} hard to compute (CDH), it is even indistinguishable from random!

Groups where DDH believed to be hard:

- Let $p = 2q + 1$ where p, q are prime. Let $\mathbb{G}$ be the subgroup of order q in $\mathbb{Z}_p^*$ [specifically, the subgroup of "quadratic residues" — $\mathbb{G} = \{h \in \mathbb{Z}_p^* : \text{there exists } x \in \mathbb{Z}_p^* \text{ where } h = x^2 \pmod p\}$]
- The set of points on an "elliptic curve" over $\mathbb{F}_p$ [will discuss in greater detail in future week]

↳ In all of these groups, the best algorithm for solving DDH is to solve discrete log (seemingly a much harder problem!)

Relationship between assumptions:

DDH $\Rightarrow$ CDH $\Rightarrow$ discrete log

strongest assumption $\longrightarrow$ weakest assumption

PKE from DDH: Let $\mathbb{G}$ be a prime order group of order p and generator g where DDH holds. Let the message space be $\mathbb{G}$.

KeyGen(1^λ): Sample $x \leftarrow \mathbb{Z}_p$ and set $h = g^x$.
Output $pk = h$ and $sk = x$.

Encrypt(pk, m): Choose $y \leftarrow \mathbb{Z}_p$. Output $ct = (g^y, h^y \cdot m)$

Decrypt(sk, ct). Write $ct = (ct_0, ct_1)$ and compute ct_1 / ct_0^{sk}

Easy to check correctness and semantic security holds under DDH

Random self-reductions: Let $\mathbb{G}$ be a group with prime order p and generator g. Suppose there exists an efficient algorithm A that solves discrete log in $\mathbb{G}$ on average:

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

Can we use A to solve discrete log in the worst-case?

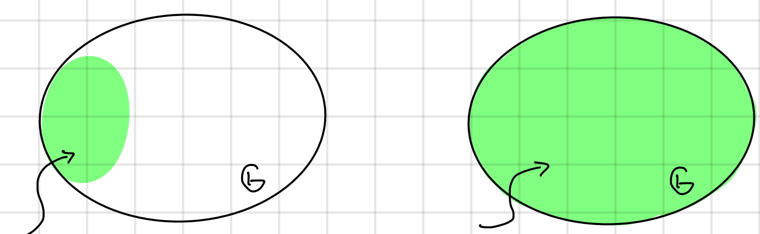
Given a discrete log challenge (g, h), choose random r and run A on (g, h^r). By construction, h^r is uniformly random, so with prob. ε, A outputs x such that $h^r = g^x$. Then $g^{xr^{-1}} = h$ so $xr^{-1} \pmod p$ is the discrete log.

↳ We have reduced solving any discrete log instance to solving a random instance of discrete log

↳ Solving random instances is as hard as solving any instance

↳ Discrete log is either hard almost everywhere or easy almost everywhere (no middle ground)

Visually:



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

Algorithm B work with prob. ε everywhere in $\mathbb{G}$

Why do we care about random self-reducibility? In cryptography, we often rely on problems that are hard on average (for randomly sampled instances). For instance, an encryption scheme secure for 90% of the keys is not useful. When a problem has a random self reduction, worst-case hardness $\Rightarrow$ average-case hardness.

PRG from DDH: Let $\mathbb{G}$ be a group of prime order p and generator g. We construct a PRG as follows:

- The description of the PRG includes a group element $h = g^x$ where $x \xleftarrow{R} \mathbb{Z}_p$
- $\text{PRG}(y) \rightarrow (g^y, g^{xy})$

Security is immediate under DDH: $(g, g^x, g^y, g^{xy}) \overset{c}{\approx} (g, g^x, g^y, g^r)$ where $r \xleftarrow{R} \mathbb{Z}_p$

Algebraic PRFs from DDH (Naor-Reingold). Let $\mathbb{G}$ be a group of prime order p and generator g. We construct a PRF $F: \mathbb{Z}_p^{n+1} \times \{0,1\}^n \rightarrow \mathbb{G}$ as follows:

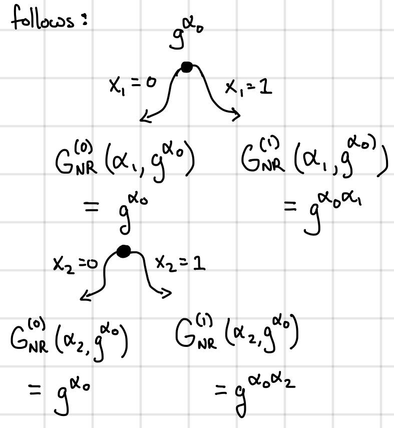
$$F((\alpha_0, \alpha_1, \ldots, \alpha_n), (x_1, \ldots, x_n)) := g^{\alpha_0 \prod_{i \in [n]} \alpha_i^{x_i}}$$

"subset product in the exponent"

<u>Security of Naor-Reingold</u>. The Naor-Reingold construction is an "augmented tree" construction. Define

$$G_{NR}(\alpha, g^\beta) \to (g^\beta, g^{\alpha\beta})$$
$$\swarrow \qquad \searrow$$
$$G_{NR}^{(0)} \qquad\qquad G_{NR}^{(1)}$$

Construction proceeds as follows:



More generally:

$F((\alpha_0, \dots, \alpha_n), x_1, \dots, x_n) :=$

$\quad t \leftarrow g^{\alpha_0}$

$\quad$ for $i = 1$ to n:

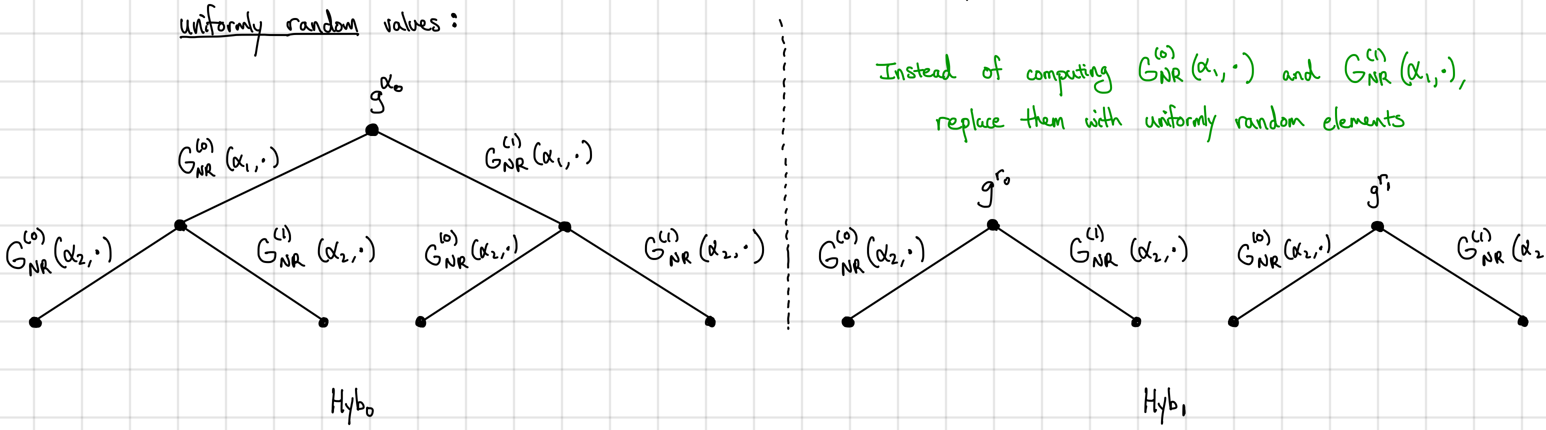
$\qquad t \leftarrow G_{NR}^{(x_i)}(\alpha_i, t)$

Suppose that for all $Q = \mathrm{poly}(\lambda)$, the following function is a secure PRG:

$$G'(\alpha_0, \alpha_1, \dots, \alpha_Q) = (G_{NR}(\alpha_0, g^{\alpha_1}), \dots, G_{NR}(\alpha_0, g^{\alpha_Q}))$$
$$= (g^{\alpha_1}, g^{\alpha_0\alpha_1}, \dots, g^{\alpha_n}, g^{\alpha_0\alpha_Q})$$

Then, the Naor-Reingold construction is a secure PRF.

<u>Proof (Sketch)</u>. We use a hybrid argument $\mathrm{Hyb}_0, \dots, \mathrm{Hyb}_n$ where evaluation in Hyb_i work by replacing first i levels of the tree with <u>uniformly random</u> values:



But... on layer n, we need to replace $2^n \neq \mathrm{poly}(\lambda)$ number of values, which does <u>not</u> follow from the above assumption!

↳ Adversary only can see <u>polynomially-many</u> outputs, so we never need to replace / simulate the entire tree, <u>only</u> the paths that the adversary queries in the PRF security game. If adversary only makes $Q = \mathrm{poly}(\lambda)$ queries, then at any level, we need to switch <u>at most</u> Q nodes from pseudorandom to truly random, which follows from our assumption.

Thus, suffice to show that G' is a secure PRG. To do so, we will rely on the DDH assumption.

<u>Claim</u>. If DDH holds in $\mathbb{G}$, then $G'(\alpha_0, \alpha_1, \dots, \alpha_Q) = (g^{\alpha_1}, g^{\alpha_0\alpha_1}, \dots, g^{\alpha_n}, g^{\alpha_0\alpha_Q})$ is a secure PRG.

Proof (Sketch) We show that if there is a distinguisher A for G', then there is an adversary B that breaks the DDH assumption.

Main challenge: Algorithm B is given a single DDH challenge (g, g^x, g^y, g^z) where $z = xy$ or $z \xleftarrow{R} \mathbb{Z}_p$ and has to simulate a PRG challenge for A. The PRG challenge should be one of two possibilities:

- Pseudorandom: $(g^{y_1}, g^{xy_1}, \dots, g^{y_n}, g^{xy_n})$ where $x, y_1, \dots, y_n \xleftarrow{R} \mathbb{Z}_p$
- Random: $(g^{y_1}, g^{z_1}, \dots, g^{y_n}, g^{z_n})$ where $y_1, \dots, y_n, z_1, \dots, z_n \xleftarrow{R} \mathbb{Z}_p$

Proof (sketch). Our goal is to take the DDH challenge and construct a PRG challenge:

$$(g, g^x, g^y, g^{xy}) \longrightarrow (g^{y_1}, g^{xy_1}, \ldots, g^{y_n}, g^{xy_n})$$
$$(g, g^x, g^y, g^z) \longrightarrow (g^{y_1}, g^{z_1}, \ldots, g^{y_n}, g^{xz_n})$$

Idea is to rely on a random self-reduction for DDH. Consider the mapping

$$(g, h, u, v) \longrightarrow (g, h, u^\alpha g^\beta, v^\alpha g^\beta) \text{ where } \alpha, \beta \xleftarrow{R} \mathbb{Z}_p$$

Suppose $(g, h, u, v) = (g, g^x, g^y, g^{xy})$ is a DDH tuple. Then,

$(g, h, u^\alpha g^\beta, v^\alpha h^\beta) = (g, g^x, g^{\alpha y + \beta}, g^{\alpha xy + \beta y})$ is still a DDH tuple and moreover $g^{\alpha y + \beta}$ is <u>uniformly random</u>!

Suppose $(g, h, u, v) = (g, g^x, g^y, g^z)$ is not a DDH tuple. Then,

$(g, h, u^\alpha g^\beta, v^\alpha h^\beta) = (g, g^x, g^{\alpha y + \beta}, g^{\alpha z + \beta x})$ is <u>not</u> a DDH tuple. Moreover $\alpha y + \beta$ and $\alpha z + \beta x$ are uniform and independent (over the choice of α, β) so $g^{\alpha y + \beta}$ and $g^{\alpha z + \beta x}$ are uniform and independent over $\mathbb{G}$!

↳ Check this! [Essentially, your argument shows that $h_{\alpha,\beta}(x) := \alpha x + \beta$ is pairwise independent if $\alpha, \beta \xleftarrow{R} \mathbb{Z}_p$.

Thus, we have a mapping that sends DDH tuples $\Rightarrow$ fresh DDH tuples and non-DDH tuples $\Rightarrow$ uniformly random values } exactly what we need to complete the above argument.

Essentially, algorithm $\mathcal{B}$ applies the random self-reduction for DDH Q-times to the DDH challenge (using independent randomness) to simulate the PRG challenge for $\mathcal{A}$.