Diffie-Hellman key exchange

- Let $\mathbb{G}$ be a group of prime order p (and generator g) – choice of group, generator, and order fixed by standard

Alice		Bob
$x \xleftarrow{R} \mathbb{Z}_p$		$y \xleftarrow{R} \mathbb{Z}_p$
	$\xrightarrow{\quad g^x \quad}$	
	$\xleftarrow{\quad g^y \quad}$	
compute $g^{xy} = (g^y)^x$		compute $g^{xy} = (g^x)^y$

shared secret: g^{xy}

But usually, we want a random <u>bit-string</u> as the key, <u>not</u> random group element

↳ Element g^{xy} has $\log p$ bits of entropy, so should be able to obtain a random bit-string with $\ell < \log p$ bits

↳ Solution is to use a "randomness extractor"

↳ Information-theoretic constructions based on universal hashing / pairwise-independent hashing (loses some bits of entropy)

↳ Use a "random oracle" or an "ideal hash function" [<u>Heuristic</u>: SHA-256 (g, g^x, g^y, g^{xy})] (very efficient in practice)

good practice to hash <u>all</u> components

[binds the key to the entire transcript]

↳ <u>Arguing security</u>: 1. Rely on HashDH assumption $(g, g^x, g^y, H(g, g^x, g^y, g^{xy})) \overset{c}{\approx} (g, g^x, g^y, r)$ where $H: \mathbb{G}^4 \to \{0,1\}^n$ and $r \xleftarrow{R} \{0,1\}^n$

2. Model H as ideal hash function $H: \mathbb{G}^4 \to \{0,1\}^n$ (i.e., random oracle) and rely on CDH in $\mathbb{G}$ [inability to evaluate H on g^{xy} $\Rightarrow$ output is random string]

<u>Public-key encryption</u>: Encryption scheme where encryption is <u>public</u> (does <u>not</u> require <u>shared secrets</u>)

- $\text{Setup}(1^\lambda) \to (pk, sk)$ [generates a public/private key-pair – also called KeyGen]
- $\text{Encrypt}(pk, m) \to c$
- $\text{Decrypt}(sk, c) \to m$

Everyone can publish a public key (in a directory)

↳ Can encrypt to anyone without exchanging keys (recipient can be <u>offline</u>)

<u>Correctness</u>: $\forall m \in \mathcal{M}: \Pr[(pk, sk) \leftarrow \text{Setup}(1^\lambda) : \text{Decrypt}(sk, \text{Encrypt}(pk, m)) = m] = 1$

<u>Security</u>: semantic security from secret-key setting, but adversary also gets public key

<u>adversary</u>		<u>challenger</u>
		$b \in \{0,1\}$
		$(pk, sk) \leftarrow \text{Setup}(1^\lambda)$
	$\xleftarrow{\quad pk \quad}$	
	$\xrightarrow{\quad m_0, m_1 \in \mathcal{M} \quad}$	
	$\xleftarrow{\quad c_b \leftarrow \text{Encrypt}(pk, m_b) \quad}$	
$b' \in \{0,1\}$		

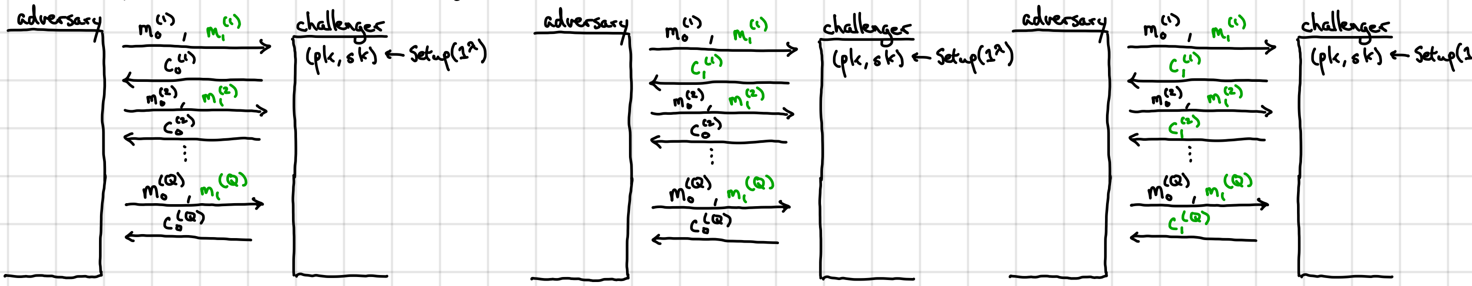
$$\text{SSAdv}[\mathcal{A}, \Pi_{PKE}] = \left|\Pr[\mathcal{A} \text{ outputs } 1 \mid b = 0] - \Pr[\mathcal{A} \text{ outputs } 1 \mid b = 1]\right|$$

In the secret-key setting, we distinguished between semantic security and CPA-security. Here, this is unnecessary since

semantic security $\Rightarrow$ CPA security [means that public-key encryption must be randomized!]

↳ Intuitively: adversary can encrypt messages on its own (using the public key)

Formally: Follows from a "hybrid" argument



$b = 0$ [always encrypt m_0] Intermediate $b = 1$ [always encrypt m_1]

Total of $Q-1$ intermediate distributions

↳ i^{th} distribution and $(i+1)^{st}$ distribution identical except on $(m_0^{(i)}, m_1^{(i)})$, challenger encrypts $m_0^{(i)}$ in distribution i and $m_1^{(i)}$ in distribution $i+1$

↳ these two distributions are indistinguishable by semantic security [in the reduction, the encryptions of the other messages (index $\neq i$) can be constructed using the public key (and do not depend on the challenger's choice bit)]

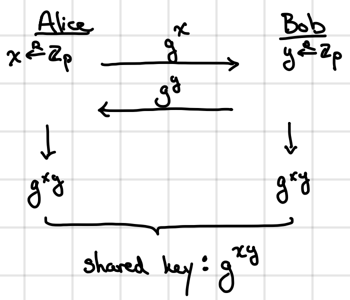
↳ if an adversary can distinguish endpoints $(b=0, b=1)$, then it must be able to distinguish a pair of intermediate distributions [by triangle inequality]

$\therefore$ semantic security $\Rightarrow$ every pair of distributions is computationally indistinguishable

$\Rightarrow$ CPA-security

PKE from DDH (ElGamal): Let $\mathbb{G}$ be a group with generator g and prime order p

Recall Diffie-Hellman key exchange:



Idea: Alice will publish $h = g^x$ as her public key

Bob encrypts by choosing fresh share g^y and uses g^{xy} to encrypt the message

security parameter dictates what group is used (eg, P-256, P-384, P-512)

$\mathrm{Setup}(1^\lambda)$: $x \xleftarrow{R} \mathbb{Z}_p$, $h \leftarrow g^x$ pk: h, sk: x $\mathcal{M} = \mathbb{G}$, $\mathcal{C} = \mathbb{G}^2$

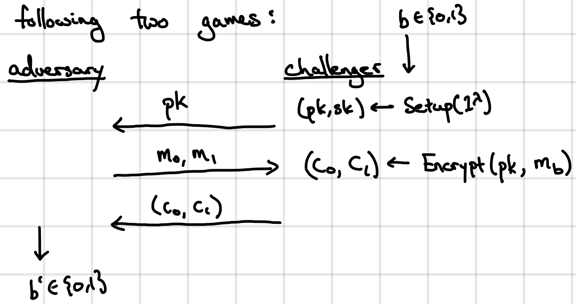
$\mathrm{Encrypt}(\overset{=h}{pk}, m)$: $y \xleftarrow{R} \mathbb{Z}_p$, $c \leftarrow (g^y, m \cdot h^y)$

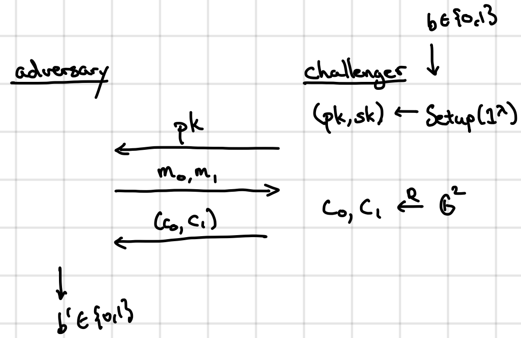
$\mathrm{Decrypt}(\overset{=x}{sk}, c)$: $m \leftarrow c_1 / c_0^x$

Correctness: $$\frac{c_1}{c_0^x} = \frac{m \cdot h^y}{(g^y)^x} = \frac{m \cdot (g^x)^y}{(g^y)^x} = \frac{m \cdot g^{xy}}{g^{xy}} = m$$

Security: If DDH holds in $\mathbb{G}$, then ElGamal is semantically secure.

Proof. Consider following two games:

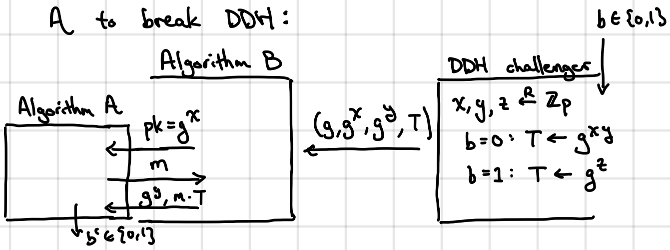




adversary's advantage in guessing b is 0 here since (c_0, c_1) is independent of (m_0, m_1)!

Claim: these two games are indistinguishable under DDH

Proof. Suppose there exists efficient $\mathcal{A}$ that can distinguish $(c_0, c_1) \leftarrow \mathrm{Encrypt}(pk, m)$ from $(c_0, c_1) \xleftarrow{R} \mathbb{G}^2$. We use $\mathcal{A}$ to break DDH:



Observe: x is uniform over $\mathbb{Z}_p$ so g^x is a properly-generated public key (for ElGamal)

if $T = g^{xy}$, then $(g^y, T \cdot m) = (g^y, g^{xy} \cdot m)$ which is the output of $\mathrm{Encrypt}(pk, m)$ with randomness y — this is exactly the distribution where $\mathcal{A}$ sees $\mathrm{Encrypt}(pk, m)$

if $T = g^z$, then $(g^y, g^z \cdot m)$ is uniform over $\mathbb{G}^2$ (since y, z are sampled independently of each other and of m) — this is exactly the distribution where $\mathcal{A}$ sees $(c_0, c_1) \xleftarrow{R} \mathbb{G}^2$

distinguishing advantage of $\mathcal{B}$ = distinguishing advantage of $\mathcal{A}$

Equivalent view: Under DDH, g^{xy} looks uniform even given g, g^x, g^y, so an ElGamal ciphertext looks indistinguishable (to an efficient adversary) from a OTP encryption

What if we want to encrypt longer messages? [or messages that is not a group element]

- Hybrid encryption (key encapsulation [KEM]):
  - Use PKE scheme to encrypt a secret key
  - Encrypt payload using secret key + authenticated encryption

called key encapsulation

$\left.\right\}$ PKE. Encrypt (pk, k) "header" [slow]
AE. Encrypt (k, m) "payload" [fast]

secret-key operations much much faster than public-key operations!

- How to derive key from group element?
  - Same as in key-exchange: hash the group element to a bit-string (symmetric key)

e.g., Hash-ElGamal: $\mathrm{Encrypt}(pk, m)$: $y \xleftarrow{R} \mathbb{Z}_p$

$$c = (g^y, m \oplus H(g, h, g^y, h^y))$$

as before, can also rely on CDH + ideal hash function (random oracle)

$H : \mathbb{G}^4 \to \{0,1\}^n$

Vanilla ElGamal described above is not CCA-secure!

Ciphertexts are malleable: given $ct = (g^y, h^y \cdot m)$, can construct ciphertext $(g^y, h^y \cdot m \cdot g)$ which decrypts to message $m \cdot g$

↳ directly implies a CCA attack

Several approaches to get CCA security from DH assumptions:

- Cramer-Shoup (CCA-security from DDH) - based on hash-proof systems
- Fujisaki-Okamoto transformation (using an ideal hash function + CDH)
- Make stronger assumption ("interactive" CDH + use ideal hash function):
  - Setup(1^λ): $x \xleftarrow{R} \mathbb{Z}_p$ $\quad$ pk: h
    $h \leftarrow g^x$ $\quad$ sk: x
  - Encrypt(pk, m): $y \xleftarrow{R} \mathbb{Z}_p$ $\quad$ $k \leftarrow H(g, g^x, g^y, h^y)$ $\quad$ $ct' \leftarrow \mathrm{Enc}_{AE}(k, m)$
    $c \leftarrow (g^y, ct')$
  - Decrypt(sk, c): $k \leftarrow H(g, g^x, c_0, c_0^x)$
    $m \leftarrow \mathrm{Dec}_{AE}(k, c_1)$

also called strong DH assumption

symmetric authenticated encryption scheme

We do not know of any groups where CDH believed to be hard, but interactive CDH is easy.

"CDH is hard even given access to a DDH oracle"

Essentially ElGamal where key derived from hash function