**CSC 591 Cryptography** 03 October 2018

# Lecture 8 – Diffie Hellman Key Exchange, DDH Assumption

*Lecturer: Alessandra Scafuro* *Scribe: Michael Suggs*

## Topic/Problem

In this discussion, we visit the notion of a *Secure Key Agreement Protocol*, as well as an example of such – most notably the *Diffie-Hellman Key Agreement.* For this, we utilize notions from abstract algebra as well as number theory to construct such a key agreement, of which an introduction to may be found in the *Preliminaries* section on page (L8 – Diffie Hellman Key Exchange, DDH Assumption-8). Security of this agreement depends on, as the name implies, the Diffie-Hellman assumption, and more specifically on one of its variants.

The Discrete-Logarithm problem is an important first step towards the aforementioned as, although not explicitly equivalent to the Diffie-Hellman problems, it is nonetheless mathematically related. Exploration of such will allow a gentler introduction to the notion of using group elements in a computational sense, and will thus pave the way for us to jump into the desired key agreement.

Lastly, by utilizing notions from one variant of Diffie-Hellman, we show how a pseudorandom generator can be constructed from such. Further, we shall touch on the notion of *one-way functions*, both in terms of their relevance and potential impact (if proven or disproven).

### Definition

**Secure Key-Agreement Protocols** The general idea behind a key agreement protocol is to develop a methodology in which two parties, of which we shall take to be the infamous *Alice* and *Bob*, can share some secret key in the explicit presence of some adversary *without* said adversary gaining any knowledge of the actual key itself. Even more so, such an exchange must be able to take place in a public setting without any eavesdropper stumbling upon, or being able to compute, the key.

The difficulty stems from the assumption that Alice and Bob have had no prior contact, and must thus establish such a key that they both agree upon in such a setting. As sending the key itself is explicitly off-limits due to the ever-present looming threat of our adversary, the key must be obfuscated in such a manner that allows for both our honest parties to recover it accurately and efficiently, though not without some insider knowledge.

As is the running theme so far, such an obfuscation can readily be obtained through the application of randomness to a desired key – though this need not be the only way. Although taking place in a public setting, the idea is to make the shared key as secure as one that was shared privately. To formalize this, we turn to the *key-exchange experiment* for adversary $\mathcal{A}$ and probabilistic protocol $\Pi$, which we may utilize to analyze the security of a given key agreement.

Key-Exchange Experiment $\mathrm{KE}^{\mathrm{eav}}_{\mathcal{A},\Pi}(n)$:

1: Alice and Bob begin with the unary security parameter $1^n$ and both execute the key-exchange protocol $\Pi$, generating a key $k$ and transcript `tsrc` of all messages sent between them.

2: A uniform $b \in \{0,1\}$ is chosen. If $b = 0$, define $\widehat{k} := k$. If instead $b = 1$, select a uniform $\widehat{k} \leftarrow \{0,1\}^n$.

3: Give both the transcript `tsrc` and unknown value $\widehat{k}$ to $\mathcal{A}$. In return, $\mathcal{A}$ will output a bit $b'$.

4: Compare. If $b' = b$, then $\mathrm{KE}^{\mathrm{eav}}_{\mathcal{A},\Pi}(n) = 1$ as $\mathcal{A}$ was able to correctly distinguish whether $\widehat{k}$ was indeed the key $k$ or uniform in $\{0,1\}^n$ from `tsrc`. If $b' \neq b$, $\mathrm{KE}^{\mathrm{eav}}_{\mathcal{A},\Pi}(n) = 0$.

$\mathrm{KE}^{\mathrm{eav}}_{\mathcal{A},\Pi}(n)$

```
1 :  $\mathrm{tscr}, k \leftarrow \Pi(\mathrm{Alice},\ \mathrm{Bob})$
2 :  $b \leftarrow \{0,1\}$
3 :  if $b = 0$ then
4 :      $\widehat{k} := k$
5 :  else
6 :      $\widehat{k} \leftarrow \{0,1\}^n$
7 :  Give tscr, $\widehat{k}$ to $\mathcal{A}$
8 :  $b' \leftarrow \mathcal{A}$
9 :  if $b' = b$ then
10 :     output 1
11 :  else
12 :     output 0
```

Figure 1: The Key-Exchange Experiment for protocol $\Pi$ and adversary $\mathcal{A}$. The leftmost gives a textual description where the rightmost gives a pseudocode depiction.

From the above, we revisit the notion of eavesdropper security, though this time for key-exchange protocols.

**Definition 1.** *A key-exchange protocol* $\Pi$ *is* secure in the presence of an eavesdropper *if, for every probabilistic polynomial-time adversary* $\mathcal{A}$*, there exists a negligible function* $\mathsf{negl}(n)$ *such that*

$$\Pr\left[KE^{eav}_{\mathcal{A},\Pi}(n) = 1\right] \leq \frac{1}{2} + \mathsf{negl}(n).$$

Observe that, although our notion of security in the presence of an eavesdropper takes a similar form, there exists a key difference – most notably that instead of observing an encryption of any messages, our adversary instead observes the exchange of a key (which indeed may later be used for encryption, but not in this explicit scenario). Thus, we refer to $\Pi$ as a *key-exchange* protocol and not as an *encryption scheme*. Further, note the wording preceding figure 1: this protocol may be probabilistic in that randomness is inherent in the secure sharing of the key itself.

## Assumption

Before delving into the realms of Diffie-Hellman, we first turn to a related (although not proven equivalent) problem – the Discrete-Logarithm problem. This problem, and the succeeding ones, depend strongly on group theoretic computations, which is touched on more heavily in the appendix. To begin, let us formalize some notation common to the following problems.

Discrete-Logarithm Experiment $\mathrm{DLog}_{\mathcal{A},\mathcal{G}}(n)$:

1: Run the group generation algorithm $\mathcal{G}$ to obtain our group description, which we write as $(\mathbb{G}, q, g)$ defined as above.

2: Select a uniform *group element* $h \in \mathbb{G}$.

3: Give the adversary $\mathcal{A}$ our group description, along with the above uniform group element as $(\mathbb{G}, q, g, h)$.

4: $\mathcal{A}$ returns some integer $x \in \mathbb{Z}_q$, which it believes to be the discrete logarithm of $h$.

5: If $h = g^x$, the output of the experiment is 1. Else, $\mathcal{A}$ has failed and the experiment outputs 0.

```
DLog_{A,G}(n)
1 :  $(\mathbb{G}, q, g) \leftarrow \mathcal{G}(1^n)$
2 :  $h \leftarrow \mathbb{G}$
// Note: $h$ is a uniform group element
3 :  $(\mathbb{G}, q, g, h) \rightarrow \mathcal{A}$
4 :  $x \leftarrow \mathcal{A}$
// Note: $x$ is an element of $\mathbb{Z}_q$
5 :  if $h = g^x$ then
6 :      $\mathrm{DLog}_{\mathcal{A},\mathbb{G}}(n)$ outputs 1
7 :  else
8 :      $\mathrm{DLog}_{\mathcal{A},\mathbb{G}}(n)$ outputs 0
```

Figure 2: The Discrete-Logarithm experiment for group generation algorithm $\mathcal{G}$ and adversary $\mathcal{A}$. The leftmost gives a textual description where the rightmost gives a pseudocode depiction.

## The Discrete-Logarithm Problem

Let $\mathcal{G}$ denote a polynomial-time *group generation algorithm* which, on an input unary security parameter, returns the *description* of a group $\mathbb{G}$. Specifically, this description of $\mathbb{G}$ details how these group elements are represented in a computational sense, though the underlying bit-strings are unnecessary for the time being. Note that, unless otherwise stated, $\mathbb{G}$ is assumed to be a cyclic group of order $q$ with generator $g \in \mathbb{G}$ such that $\langle g \rangle = \mathbb{G}$.

As $g$ generates all of $\mathbb{G}$, no more and no less, we may infer that for some other group element $h \in \mathbb{G}$, there exists a unique number of applications of $g$ to itself to calculate $h$. Formally, we have for $h \in \mathbb{G}$, there exists some unique $x \in \mathbb{Z}_q$ such that $g^x = h$ – the reason we select an integer from $\mathbb{Z}_q$ is directly related to the order of $\mathbb{G}$ being $q$, and thus $\mathbb{G} = \{g^0, g^1, g^2, \ldots, g^{q-3}, g^{q-2}, g^{q-1}\}$. We call such an $x$ the *discrete logarithm of $h$ with respect to $g$*, as by properties of logarithms we may rewrite $g^x = h$ as $\log_g(h) = x$, with the logarithm being only defined for a finite (or *discrete*) range of values as defined. That said, our discrete logarithm obeys all of your familiar rules of logarithms and thus may be, for the most part, treated as such.

As with our presentation of key-exchanges, we now examine the *discrete-logarithm experiment* in figure 2. Similar to previous experiments, we also define this in terms of probability and negligible functions – though this time with respect to the *hardness* of computing the discrete logarithm. We thus have as follows:

**Definition 2.** *Given a group generation algorithm $\mathcal{G}$ outputting the description of a cyclic group $\mathbb{G}$, we say that the* DISCRETE-LOGARITHM *problem is* hard relative to $\mathcal{G}$ *if, for all probabilistic polynomial-time adversaries $\mathcal{A}$ there exists a negligible function* $\mathsf{negl}(n)$ *such that*

$$\Pr\left[DLog_{\mathcal{A},\mathcal{G}}(n) = 1\right] \leq \mathsf{negl}(n)\,.$$

At a high level, the underlying hardness of this problem is that taking the discrete logarithm of a group element in some cyclic group is inherently hard – especially for groups of large order. Specifically, it is speculated that the Discrete-Logarithm problem is a member of the complexity class NP-intermediate, and thus belongs neither in P nor NP-complete. However, given a quantum adversary , the Discrete-Logarithm problem is *decidable* in polynomial-time (with bounded-error), and thus belongs within the complexity class BQP[1].

### The Diffie-Hellman Problem

As stated prior, it is yet unknown if there is an equivalency between the Discrete-Logarithm problem and that of the Diffie-Hellman problem – however, it is clear that the two are quite mathematically related.

Again, we take a cyclic group $\mathbb{G}$ of order $q$ with generator $g$ such that $\langle g \rangle = \mathbb{G}$ as our basis. However, to compound the complexity of taking discrete logarithms, Diffie-Hellman instead requires

$$\mathrm{DH}_g(h_1, h_2) = g^{\log_g(h_1) \cdot \log_g(h_2)},$$

for $h_1, h_2 \in \mathbb{G}$. As stated in the introduction, there are two quite relevant variations on this assertion that are increasingly hard to compute – namely the *computational* and *decisional* Diffie-Hellman problems, denoted *CDH* and *DDH* respectively.

### Computational Diffie-Hellman and Decisional Diffie-Hellman

As the name implies, CDH requires some adversary to compute the value of $\mathrm{DH}_g(h_1, h_2)$ as defined above for *uniform elements* $h_1$ and $h_2$ in $\mathbb{G}$. Security of CDH depends directly on the hardness of taking discrete logarithms with respect to $\mathcal{G}$ – if this were not to hold, computing CDH would be quite easy indeed. For example:

**Example 3.** *Assume that taking discrete logarithms is* ***easy with respect to some*** $\mathcal{G}$*. Thus, for* $\langle g \rangle = \mathbb{G}$ *where* $|\mathbb{G}| = q$*, we may select some* $a, b \in \mathbb{Z}_q$ *and compute* $g^{ab}$ *as follows:*

1. *Take the discrete logarithm with respect to* $g$ *of* $g^a$*, giving* $a = \log_g(g^a)$*.*
2. *As we are given* $g^b$*, compute* $g^{ab} = (g^b)^a$*, thus solving the CDH problem for* $\mathcal{G}$*.*

◇

Due to the above example, we see the necessitation of hardness with respect to $\mathcal{G}$ is not one to be taken lightly.

The Decisional Diffie-Hellman problem presents a stronger notion of both complexity and security, as instead of asking some adversary to compute the value of $\mathrm{DH}_g(h_1, h_2)$, we instead require for our adversary to *decide* whether that which they have been presented is indeed $\mathrm{DH}_g(h_1, h_2)$ or some uniform group element $g^z$ for $z \in \mathbb{Z}_q$. Specifically, we say the following:

**Definition 4.** *The DDH problem is* hard with respect to $\mathcal{G}$ *if for all probabilistic polynomial-time adversaries* $\mathcal{A}$ *there exists a negligible function* $\mathsf{negl}(n)$ *such that*

$$|\Pr[\mathcal{A}(\mathbb{G}, q, g, g^x, g^y, g^z) = 1] - \Pr[\mathcal{A}(\mathbb{G}, q, g, g^x, g^y, g^{xy}) = 1]| \leq \mathsf{negl}(n).$$

*Where* $\mathcal{G}(1^n) \to (\mathbb{G}, q, g)$ *and* $x, y, z \leftarrow Z_q$ *are uniform in* $\mathbb{Z}_q$*.*

[1]https://arxiv.org/abs/quant-ph/9508027

```
Decisional Diffie-Hellman Experiment $\mathrm{DDH}_{\mathcal{A},\mathcal{G}}(n)$:
1: Run the group generation algorithm $\mathcal{G}$ to obtain our group description,
which we write as $(\mathbb{G}, q, g)$ defined as above.
2: Select uniform $x, y, z \in \mathbb{Z}_q$ and compute $g^x, g^y$.
3: Set $h_0 = (g^x)^y$, and $h_1 = g^z$.
4: Flip bit $b \in \{0, 1\}$.
5: Give the adversary $\mathcal{A}$ all the above as $(\mathbb{G}, q, g, g^x, g^y, h_b)$.
6: $\mathcal{A}$ outputs bit $b'$.
7: If $b' = b$, the output of the experiment is 1. Else, $b' \neq b$ and the output
of the experiment is 0.
```

Figure 3: The DDH experiment for group generation algorithm $\mathcal{G}$ and adversary $\mathcal{A}$. The leftmost gives a textual description where the rightmost gives a pseudocode depiction.

## Scheme

Continuing from the above description of the Diffie-Hellman problems, we recall that mathematically hard problems often make excellent bases for cryptographic constructions – thus, enter the *Diffie-Hellman Key Agreement.* Based on the hardness of the Decisional Diffie-Hellman problem, the key agreement itself takes a remarkably similar approach, which we visualize for clarity as follows: To break this down, we first note the similarities to

$$\boxed{\begin{array}{lcl} \textbf{Alice} & & \textbf{Bob} \\ (\mathbb{G}, q, g) \leftarrow \mathcal{G}(1^n) & & \\ x \leftarrow \mathbb{Z}_q & & \\ h_A := g^x & & \\ & \xrightarrow{\mathbb{G}, q, g, h_A} & \\ & & y \leftarrow \mathbb{Z}_q \\ & & h_B := g^y \\ & \xleftarrow{h_B} & \\ k_A := h_B^x & & k_B := h_A^y \end{array}}$$

Figure 4: A visualization of the Diffie-Hellman Key Agreement.

the DDH problem as described before. Specifically, we have a group generation algorithm that outputs a description of some cyclic group $\mathbb{G}$ of order $q$ with generator $g$ such that $\langle g \rangle = \mathbb{G}$. Alice then uniformly selects some integer $x$ from the group of integers modulo $q$, which is used to compute $h_A := g^x$. Note, to compute $x$ given $h_A$, one must thus compute the discrete logarithm of $h_A$ with respect to $g$ (since $\log_g(h_A) = x$). Alice then sends the group along with its order, the generator, and the computed element $h_A$ to Bob for further computation.

Upon receiving the above from Alice, Bob begins by selecting a uniform integer $y$ from $\mathbb{Z}_q$, which is then used to compute Bob's $h_B := g^y$. Note that both $h_A$ and $h_B$ utilize uniform integers, which represents the introduction of randomness to our agreement. Bob then sends $h_B$ to Alice, after which they output a value – Alice outputs $k_A := h_B^x$ and Bob outputs $k_B := h_A^y$. If our definition of a key agreement holds, we see that $k_A = k_B = k$ and some adversary given the entire exchange between Alice and Bob cannot distinguish between $k$ and some uniform string.

First and foremost, we must convince ourselves of the correctness of the aforementioned. Note that each party introduced their own randomness into the scheme, which was then passed on to each other in a clearly decipherable manner – although not by itself. Instead, the randomness was "hidden" by using it as an exponent for the group generator $g$. Since each party only knows their own randomness, how can they be sure that the key remaining at the end is truly the correct key?

Observe how the randomness of each party is utilized – Alice computes $h_A := g^x$ with her uniform $x \in \mathbb{Z}_q$ and Bob computes $h_B := g^y$ with his uniform $y \in \mathbb{Z}_q$. At the end, each party takes the other party's value of $h$ and applies their own randomness to it – this combined applications of randomness result in the shared key as follows:

$$k_A = h_B^x = (g^y)^x = g^{xy} = (g^x)^y = h_A^y = k_B.$$

By following the above protocol, we see that both Alice and Bob have successfully communicated a key via group elements by keeping a bit of randomness for themselves in addition to that applied to the generator to obtain each respective $h$.

## Security Proof

As the name implies, the security of the Diffie-Hellman Key Exchange is directly related to the computational hardness of the Decisional Diffie-Hellman problem – recall that, as DDH gives a stronger notion of security than CDH, the utilization of such thus implies the possibility stronger notions of security to the agreement as a whole. Recalling out key agreement experiment $\mathrm{KE}^{\mathrm{eav}}_{\mathcal{A},\Pi}(n)$, we modify this scheme slightly to the scheme $\mathrm{KE}^{\mathrm{eav}}_{\mathcal{A},\Pi}(n)$ as is found in figure 5, which simply substitues the selection of a uniform bit-string when $b = 1$ with that of a uniform group element instead.

The following proof show that, in order to have an adversary $\mathcal{A}$ that distinguishes $k = k_A = k_B$ from a uniform element in $\mathbb{G}$, we must therefore have an adversary to solve the decisional Diffie-Hellman problem given as the exact same $\mathcal{A}$. Instead of splitting into explicit cases, the proof provided here follows the format as seen in the textbook, which combines the two cases – $\widehat{k} = g^{xy}$ for $b = 0$ and $\widehat{k} = g^z$ for $b = 1$ – into conditional probabilities on the key experiment itself. To formalize the goal for our proof, we note the following theorem.

**Theorem 5.** *If the decisional Diffie-Hellman problem is hard with respect to $\mathcal{G}$, then the Diffie-Hellman key-exchange protocol $\Pi$ is secure in the presence of an eavesdropper $\mathcal{A}$.*

**Proof:** Let $\mathcal{A}$ be some probabilistic polynomial-time adversary in the experiment $\widehat{\mathrm{KE}}^{\mathrm{eav}}_{\mathcal{A},\Pi}(n)$. On initialization, $\mathcal{A}$ receives $(\mathbb{G}, q, g, h_A, h_B, \widehat{k})$, where $\widehat{k}$ may either be the actual key $k =$

```
$\widehat{\mathrm{KE}}^{\mathrm{eav}}_{\mathcal{A},\Pi}(n)$

1 :  $\mathrm{tscr}, k \leftarrow \Pi(\mathrm{Alice},\ \mathrm{Bob})$
2 :  $b \leftarrow \{0,1\}$
3 :  if $b = 0$ then
4 :    $\widehat{k} := k$
5 :  else
6 :    $\widehat{k} \leftarrow \mathbb{G}$
7 :  Give tscr, $\widehat{k}$ to $\mathcal{A}$
8 :  $b' \leftarrow \mathcal{A}$
9 :  if $b' = b$ then
10 :    output 1
11 :  else
12 :    output 0
```

Figure 5: A modified key-agreement experiment for the Diffie-Hellman Key Exchange. The only difference is that, instead of selecting a uniform bit-string on line 6, we instead select a uniform $\widehat{k}$ from out cyclic group $\mathbb{G}$.

$k_A = k_B$ or a uniform group element. Further, assume that the Decisional Diffie-Hellman problem is hard with respect to $\mathcal{G}$ and thus we have that there exists a negligible function $\mathsf{negl}(n)$ such that

$$|\Pr[\mathcal{A}(\mathbb{G}, q, g, g^x, g^y, g^z) = 1] - \Pr[\mathcal{A}(\mathbb{G}, q, g, g^x, g^y, g^{xy}) = 1]| \leq \mathsf{negl}(n)\,.$$

We show that deciding between the two cases above is equivalent to solving the Decisional Diffie-Hellman problem with respect to $\mathcal{G}$.

$$\begin{aligned}
\Pr\left[\widehat{KE}^{\mathrm{eav}}_{\mathcal{A},\Pi} = 1\right] &= \frac{1}{2}\Pr\left[\widehat{KE}^{\mathrm{eav}}_{\mathcal{A},\Pi} = 1 \;\middle|\; b = 0\right] + \frac{1}{2}\Pr\left[\widehat{KE}^{\mathrm{eav}}_{\mathcal{A},\Pi} = 1 \;\middle|\; b = 1\right] \\
&= \frac{1}{2}\Pr[\mathcal{A}(\mathbb{G}, q, g, g^x, g^y, g^{xy}) = 0] + \frac{1}{2}\Pr[\mathcal{A}(\mathbb{G}, q, g, g^x, g^y, g^z) = 1] \\
&= \frac{1}{2}(1 - \Pr[\mathcal{A}(\mathbb{G}, q, g, g^x, g^y, g^{xy}) = 1]) + \frac{1}{2}\Pr[\mathcal{A}(\mathbb{G}, q, g, g^x, g^y, g^z) = 1] \\
&= \frac{1}{2} + \frac{1}{2}(\Pr[\mathcal{A}(\mathbb{G}, q, g, g^x, g^y, g^z) = 1] - \Pr[\mathcal{A}(\mathbb{G}, q, g, g^x, g^y, g^{xy}) = 1]) \\
&\leq \frac{1}{2} + \frac{1}{2}|\Pr[\mathcal{A}(\mathbb{G}, q, g, g^x, g^y, g^z) = 1] - \Pr[\mathcal{A}(\mathbb{G}, q, g, g^x, g^y, g^{xy}) = 1]|\,. \\
&\leq \frac{1}{2} + \frac{1}{2}\cdot\mathsf{negl}(n)\,.
\end{aligned}$$

We thus observe that, since $\widehat{KE}^{\mathrm{eav}}_{\mathcal{A},\Pi}$ outputs 1 with probability less than negligible, our assumption on the hardness of DDH with respect to $\mathcal{G}$ holds. This therefore implies that $\Pi$ is secure, as desired.

$\square$

To understand more intuitively, the probability that $\widehat{\mathrm{KE}}^{\mathrm{eav}}_{\mathcal{A},\Pi}$ outputs 1 is precisely when $\mathcal{A}$ can distinguish between $g^{xy}$ and $g^z$ and therefore breaking $\Pi$. We therefore consider the probabilities in either case (when $b = 0$ and $b = 1$ respectively), splitting into two representations with the group elements written out to denote instances of the experiment $\mathrm{DDH}_{\mathcal{A},\mathcal{G}}(n)$. In this, we see that when $b = 0$ in $\mathrm{DDH}_{\mathcal{A},\mathcal{G}}(n)$, $\mathcal{A}$ is presented with $(\mathbb{G}, q, g, g^x, g^y, g^{xy}$ – similarly, when $b = 1$ in $\mathrm{DDH}_{\mathcal{A},\mathcal{G}}(n)$, $\mathcal{A}$ is presented with $(\mathbb{G}, q, g, g^x, g^y, g^z)$. Thus, by plugging in the probabilities (and making them align by taking the inverses), we conclude that, as the probability that $\widehat{\mathrm{KE}}^{\mathrm{eav}}_{\mathcal{A},\Pi} = 1$ for either value of $b$ is less than $\mathsf{negl}(n)$, the Diffie-Hellman key agreement $\Pi$ is thus secure.

## Preliminaries[2]

### Number Theory

Before we discuss groups and their related topics, we first shall turn to a few essential notions from number theory – specifically, that of *modular arithmetic*. Intrinsic to this is the concept of a modulus, which in turn relies on the existence of divisibility.

**Definition 6** (Divisible)**.** *Given two integers $x, y \in \mathbb{Z}$, we say that $x$* divides *$y$ if, for some $k \in \mathbb{Z}$, it holds that $y = xk$. We denote this as $x \mid y$ and say that $x$ is a divisor of $y$. Further, we note that $y$ is a multiple of $x$.*

To apply this towards modular arithmetic we turn to the *division algorithm*, which allows us to define divisibility relations between numbers that are not explicit divisors or multiples of one another. We neglect proof for this claim, as such is outside the scope of our present interests.

**Definition 7** (Division Algorithm)**.** *For integers $a, b \in \mathbb{Z}$ with $b > 0$, there exists $q, r \in \mathbb{Z}$ such that $a = bq + r$. We say that $r$ is the* remainder *and $q$ is the* quotient *of division of $a$ by $b$. If $r = 0$, then we say that $b \mid a$.*

From the above, we may directly rearrange this to see that this implies $(a - r) = bq$, or perhaps more usefully that $b \mid (a - r)$. We say that $a$ is *congruent to $r$ modulo $b$* and write this as $a \equiv r \bmod b$ (or equivalently as $a \equiv_b r$). For future discussion, it is worth noting that the congruence operation ($\equiv$) is an equivalence relation.

**Definition 8** (Modulo)**.** *Let $a, b, m \in \mathbb{Z}$ be integers. If $m \mid (a - b)$, then we say that $a$ is congruent to $b$ modulo $m$, and denote this as $a \equiv_m b$, or equivalently as $a \equiv b$ modulo $m$.*

From this, we arrive at the notion of a *quadratic residue*, or simply a *residue modulo $n$*. The set of residues modulo $n$ is defined as all values of $y$ which satisfy the congruence $y \equiv x^2 \bmod n$.

**Example 9.** *Let $n = 6$. We define the residues of 6 as the set $\{0, 1, 3, 4\}$ as follows:*

$$\begin{array}{ll} 0^2 \equiv_6 0 & 3^2 \equiv_6 3 \\ 1^2 \equiv_6 1 & 4^2 \equiv_6 4 \\ 2^2 \equiv_6 4 & 5^2 \equiv_6 1 \end{array}$$

[2]For a more exhaustive presentation of elementary group theory, please see Chapter 8.1 in our textbook. Only the essentials for the stated problems is covered in this section.

◇

**Prime numbers** Before venturing into the realm of group theory, we make one last note on a specific classification of numbers of particular interest in number theory (and thus cryptography). In general, the majority of numbers encountered are `composite` – numbers that may be represented as the product of other numbers, aside from the trivial case of 1 and itself. On the other hand, we consider *prime numbers* – numbers greater than 1 in which the only positive integer divisors are 1 and itself.

Aside from prime numbers, we may consider two numbers that are *relatively prime* or *coprime* to each other, in which their greatest common divisor is 1. Further, we make note of *Euler's totient function* as $\phi(n)$ which, for some $n \in \mathbb{Z}^+$, returns the number of positive integers relatively prime to $n$ that are also at most $n$.

## Group Theory

We begin by describing what exactly a *group* is. Simply put, we may envision a group $\mathbb{G}$ as being a set of elements together with some operation satisfying four properties. For our purposes, the elements in question are taken to be real numbers. Further, we may assume the *group operation* to be either multiplication or addition, which we shall denote $\mathbb{G}^*$ or $\mathbb{G}^+$ respectively, for some arbitrary group $\mathbb{G}$.

Put more succinctly, we have the following definition:

**Definition 10** (Group)**.** *Given a set of elements $X$ and a binary operation $(\cdot)$, we may define a group $\mathbb{G}(X, \cdot)$ (or more simply $\mathbb{G}$, as consisting of the elements of $X$ under $(\cdot)$ if the following hold:*

1. *Closure: For all $x_i, x_j \in \mathbb{G}$, then $x_i \cdot x_j \in \mathbb{G}$.*
2. *Associativity: For all $x_i, x_j, x_k \in \mathbb{G}$, it holds that $(x_i \cdot x_j) \cdot x_k = x_i \cdot (x_j \cdot x_k) \in \mathbb{G}$.*
3. *Identity: There exists some element $e \in \mathbb{G}$ such that $x \cdot e = e \cdot x = x \in \mathbb{G}$.*
4. *Inverse: For all $x_i \in \mathbb{G}$, there exists $x_j \in \mathbb{G}$ such that $x_i \cdot x_j = x_j \cdot x_i = e \in \mathbb{G}$.*

**Modulo groups** Returning to modular arithmetic, it naturally follows that we may define a group over some subset of the integers modulo some integer $n$. In the additive sense, we may define such a group as being over the set $\{0, 1, \ldots, n-1\}$ – we leave verification of such as an exercise. However, of particular interest for our discussion is such a group under multiplication – thus, for the remainder of this paper consider all groups to be taken as multiplicative groups.

**Definition 11.** *A group $\mathbb{G}$ is said to be a* **multiplicative group of integers modulo** $n$ *if it is defined as a group under multiplication over the set of residues modulo $n$. Thus, the underlying set composing $\mathbb{G}$ is the set of all residues of $n$, of which there are $\phi(n)$. As $\mathbb{G}$ is a group, all the aforementioned group properties must also hold.*

**Generators and cyclic groups** For a given group $\mathbb{G}$, we define a *generator* of $\mathbb{G}$ as being an element $g \in \mathbb{G}$ if the set resulting from repeated applications of the group operation of $g$ to itself is exactly $\mathbb{G}$. Taking $g^i = \overbrace{g \cdot g \cdots g}^{i}$, we have $\langle g \rangle = \{g^0, g^1, \ldots, g^n\} = \mathbb{G}$ for a group $\mathbb{G}$ of order $n$ with $g \in \mathbb{G}$.

**Definition 12** (Generator). *Let $\mathbb{G}(X, \cdot)$ be a group and $g \in \mathbb{G}$ be an element of $\mathbb{G}$. We say $g$ is a* generator *of $\mathbb{G}$ if, for all $q \in \mathbb{G}$, there exists some $i \in \mathbb{Z}$ such that $g^i = q$. We denote such a generator as $\langle g \rangle$ and say that $g$ generates $\mathbb{G}$, or $\langle g \rangle = \mathbb{G}$.*

**Definition 13** (Cyclic group). *We say a group $\mathbb{G}$ is* cyclic *if there exists some element $g \in \mathbb{G}$ that generates all of $\mathbb{G}$.*