weaken RSA in any way. The decryption exponents (the private keys) of different users will be different, since different p, q, and consequently N are chosen for each key pair.

Amazingly, a suitable choice for the common encryption exponent is $e = 3$. With this choice of e, each public key encryption only requires two multiplications. However, the private key operations remain expensive since there is no special structure for d. This is often acceptable since many encryptions may be done by a central server, while the decryption is effectively distributed among the clients. Of course, if the server needs to compute digital signatures, then a small e does not reduce its workload. Although the math would work, it would certainly be a bad idea to choose a common value of d for all users.

With an encryption exponent of $e = 3$, the following *cube root attack* is possible. If the plaintext M satisfies $M < N^{1/3}$, then $C = M^e = M^3$, that is, the mod N operation has no effect. As a result, an attacker can simply compute the usual cube root of C to obtain M. In practice, this is easily avoided by padding M with enough bits so that, as a number, $M > N^{1/3}$.

If multiple users all have $e = 3$ as their encryption exponent, another type of the cube root attack exists. If the same message M is encrypted with three different users' public keys, yielding, say, ciphertext C_0, C_1, and C_2, then the Chinese Remainder Theorem [43] can be used to recover the message M. This is also easily avoided in practice by randomly padding each message M or by including some user-specific information in each M, so that the messages actually differ.

Another popular common encryption exponents is $e = 2^{16}+1$. With this e, each encryption requires only 17 steps of the repeated squaring algorithm. An advantage of $e = 2^{16} + 1$ is that the same encrypted message must be sent to $2^{16} + 1$ users before the Chinese Remainder Theorem attack can succeed.

Next, we'll examine the Diffie-Hellman key exchange algorithm, which is a very different sort of public key algorithm. Whereas RSA relies on the difficulty of factoring, Diffie-Hellman is based on the so-called discrete log problem.

4.4 Diffie-Hellman

The Diffie-Hellman key exchange algorithm, or DH for short, was invented by Malcolm Williamson of GCHQ and shortly thereafter it was independently reinvented by its namesakes, Whitfield Diffie and Martin Hellman [191].

The version of DH that we discuss here is a key exchange algorithm because it can only be used to establish a shared secret. The resulting shared secret is generally used as a shared symmetric key. It's worth emphasizing that, in this book, the words "Diffie-Hellman" and "key exchange" always go together—DH is not for encrypting or signing, but instead it allows users to establish a shared symmetric key. This is no mean feat, since this key establishment problem is one of the fundamental problems in symmetric key cryptography.

The security of DH relies on the computational difficulty of the *discrete log* problem. Suppose you are given g and $x = g^k$. Then to determine k you would compute the logarithm, $\log_g(x)$. Now given g, p, and $g^k \bmod p$, the problem of finding k is analogous to the logarithm problem, but in a discrete setting. This discrete version of the logarithm problem is, not surprisingly, known as the discrete log problem. As far as is known, the discrete log problem is very difficult to solve, although, as with factoring, it is not known to be, say, NP-complete.

The mathematical setup for DH is relatively simple. Let p be prime and let g be a *generator*, which means that for any $x \in \{1, 2, \ldots, p-1\}$ there exists an exponent n such that $x = g^n \bmod p$. The prime p and the generator g are public.

For the actual key exchange, Alice randomly selects a secret exponent a and Bob randomly selects a secret exponent b. Alice computes $g^a \bmod p$ and sends the result to Bob, and Bob computes $g^b \bmod p$ and sends the result to Alice. Then Alice computes

$$(g^b)^a \bmod p = g^{ab} \bmod p$$

and Bob computes

$$(g^a)^b \bmod p = g^{ab} \bmod p$$

and $g^{ab} \bmod p$ is the shared secret, which is typically used as a symmetric key. The DH key exchange is illustrated in Figure 4.1.

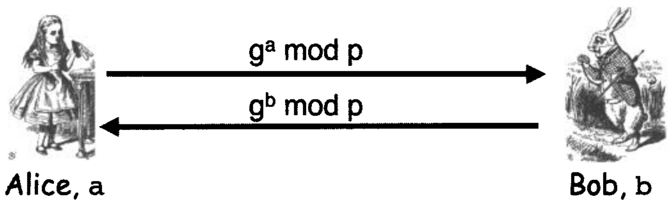

Figure 4.1: Diffie-Hellman Key Exchange

The attacker Trudy can see $g^a \bmod p$ and $g^b \bmod p$, and it seems that she is tantalizingly close to knowing the secret $g^{ab} \bmod p$. However,

$$g^a \cdot g^b = g^{a+b} \neq g^{ab} \bmod p$$

Apparently, Trudy needs to find either a or b, which appears to require that she solve a difficult discrete log problem. Of course, if Trudy can find a or b

or $g^{ab} \bmod p$ by any other means, the system is broken. But, as far as is known, the only way to break DH is to solve the discrete log problem.

There is a fundamental problem with the DH algorithm—it is susceptible to a man-in-the-middle, or MiM, attack.[6] This is an active attack where Trudy places herself between Alice and Bob and captures messages from Alice to Bob and vice versa. With Trudy thusly placed, the DH exchange can be easily subverted. In the process, Trudy establishes a shared secret, say, $g^{at} \bmod p$ with Alice, and another shared secret $g^{bt} \bmod p$ with Bob, as illustrated in Figure 4.2. Neither Alice nor Bob has any clue that anything is amiss, yet Trudy is able to read or change any messages passing between Alice and Bob.[7]

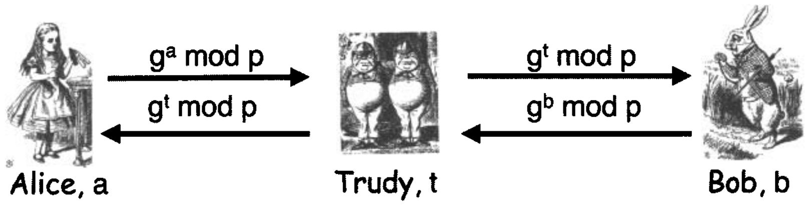

Figure 4.2: Diffie-Hellman Man-in-the-Middle Attack

The MiM attack in Figure 4.2 is a serious concern when using DH. There are several possible ways to prevent the attack, including the following:

1. Encrypt the DH exchange with a shared symmetric key.
2. Encrypt the DH exchange with public keys.
3. Sign the DH values with private keys.

At this point, you should be baffled. After all, why would we need to use DH to establish a symmetric key if we already have a shared symmetric key (as in 1) or a public key pair (as in 2 and 3)? This is an excellent question to which we'll give an excellent answer when we discuss protocols in Chapters 9 and 10.

4.5 Elliptic Curve Cryptography

Elliptic curves provide an alternative domain for performing the complex mathematical operations required in public key cryptography. So, for example, there is an elliptic curve version of Diffie-Hellman.

[6]Your politically incorrect author refuses to use the term "middleperson" attack.

[7]The underlying problem here is that the participants are not authenticated. In this example, Alice does not know she's talking to Bob and vice versa. It will be a few more chapters before we discuss authentication protocols.