

Southeast Region Research Initiative

A project of the Department of Homeland Security

Cybersecurity

Workshop 2

Public Key Cryptography

Robert Hochberg
East Carolina University



This investigation was supported by Department of Energy agreement.

Day 2 - Public Key Cryptography

I. Introduction

- A. Letter-by-letter encryption
 1. Here we encrypt a message by encrypting each individual character in the message
 2. Some schemes include
 - a. replace letter i with letter $i + 1$
 - b. replace letter i with letter $i \times 13 \bmod 96$
 3. All such schemes are easily decrypted, however, by such tricks as examining the frequency of each character in the encrypted message and comparing with the expected frequency of characters in a typical English message

II. Block Encryption and One-Way Functions

- A. It is more secure to break messages into blocks of some fixed size and encrypt whole blocks at a time.
- B. It is infeasible to construct a lookup-table containing all blocks, if the block size is sufficiently large. For example, if you are encrypting blocks of bits with blocks of size k , then there would be 2^k blocks in the lookup table. And this is large even for relatively small k .
- C. Modern encryption schemes encrypt messages in blocks
- D. A one-way function is a function that is relatively easy to compute but difficult to invert. The two schemes mentioned in the introduction are easy to “undo,” and are therefore not good for encryption
- E. The problem of factoring is difficult, however. For example, if you were given two 1000-digit prime numbers, it would not be hard for a computer to compute their product. It could easily do so in under a second. But factoring the resulting product would take millions of years with the mathematics and technology available today.
- F. The difficulty of factoring makes it an ideal basis for building secure encryption schemes.

III. Public Key Encryption

- A. With public key encryption an entity who wishes to receive encrypted messages publishes a public key which any sender may use to encrypt the message, with the expectation that only the recipient can decrypt the message.
- B. The algorithm used to encrypt is also considered public. Thus the public has complete information about how a message has been encrypted, but is still unable to decrypt it.
- C. The recipient maintains a private key which is used to decrypt.
- D. The RSA algorithm uses old ideas in number theory, together with a novel algorithm, to encrypt
 1. It is based on the fact that multiplication of two integers is quick, but factoring integers can be difficult
 2. For example, a computer can multiply two thousand-digit prime numbers together in under a second, but factoring that product (without knowing the factors, of course) would presently take millions of years on millions of computers.

IV. Key Exchange and Digital Signatures

- A. Secret keys may also be shared in a public way. In such a scheme two parties each select a prime number which they keep secret, but exchange messages which are considered public. In the end they will have arrived at some secret key which they both know, but any eavesdropper witnessing all of the messages will still not be able to deduce the key itself.
- B. One example of such a public key-exchange algorithm is the Diffie-Hellman key exchange algorithm, using mathematics similar to that of RSA
- C. Digital signatures allow a user to “sign” a message, M with some signature, S , which is a function M
 1. When a recipient receives M and S , he can verify that M is the message that the sender intended to send, and that it has not been tampered with in transit.
 2. Digital signatures also allow a recipient to be sure that the sender can not repudiate the document later (deny that he sent it)

Public Key Encryption

Cleverness University

Criminal Events Division

Please encrypt all correspondence using the following encryption key:

“1789234987123466715491694383489”

In this scenario, the algorithm used to encrypt messages, as well as the key given in the message, is publicly available. Does this necessarily compromise the security of the encryption scheme?

In a public key encryption scheme, the recipient wishes to publish the encryption key and encryption algorithm, so that anyone may send encrypted messages, but only the recipient may decrypt.

ASCII

The ASCII system is used to encode text messages as sequences of numbers, a first step for cryptosystems!

32		48	0	64	@	80	P	96	`	112	p
33	!	49	1	65	A	81	Q	97	a	113	q
34	"	50	2	66	B	82	R	98	b	114	r
35	#	51	3	67	C	83	S	99	c	115	s
36	\$	52	4	68	D	84	T	100	d	116	t
37	%	53	5	69	E	85	U	101	e	117	u
38	&	54	6	70	F	86	V	102	f	118	v
39	'	55	7	71	G	87	W	103	g	119	w
40	(56	8	72	H	88	X	104	h	120	x
41)	57	9	73	I	89	Y	105	i	121	y
42	0	58	:	74	J	90	Z	106	j	122	z
43	0	59	;	75	K	91	[107	k	123	{
44	,	60	<	76	L	92	\	108	l	124	
45	-	61	0	77	M	93]	109	m	125	}
46	.	62	>	78	N	94	^	110	n	126	~
47	/	63	?	79	O	95	_	111	o	127	_

Scheme A — Add One

In this scheme, we replace each “letter” in a message with the “letter” that comes after it, where we use the ASCII system of letters and digits:

Thus, the message:

Brice lives at 10 Oak Street

becomes:

Csjdf!mjwft!bu!21!Pbl!Tusffu

Although this message is hard to read, it is not hard to decode.

Q: What is the decoding algorithm?

A: Subtract one.

Decode This

Trvfbnjti!Pttjgsbhf

These words appeared in a famous Challenge message from RSA. The RSA company, which sells encryption software and services, published in 1977 an encrypted version of the phrase "*The magic words are squeamish ossifrage*" and challenged the world to decode it.

The idea was to show how secure their encryption algorithm was. At the time, they predicted it would take 40,000,000,000,000,000 years to decipher.

It was decrypted in 1994.

In the intervening time, computers had become much quicker, the Internet made it possible for thousands of computers to work on the problem together, and, most significantly, mathematical factoring algorithms improved.

Scheme 2 — $[(x - 32) \times 13] \bmod 96 + 32$

In this scheme, we subtract 32 from our ASCII code, multiply by 13, take the remainder when dividing by 96, and add back the 32. (In this example, 13 would be considered the "key.")

(The reason for subtracting and adding 32 is to force the answers to lie in the range 32-127. The characters below 32 and above 127 are generally not printable.)

For example,

Brice lives at 10 Oak Street

becomes

Z*u'A <u^A7 mD =0 Cm/ wD*AAD

When encrypted by Scheme 2.

Decryption in Scheme 2

It is easy to undo the addition and subtraction of 32. But how can we undo multiplication by 13 and taking the remainder mod 96?

As we saw yesterday, we can do "division" in modular arithmetic if we can find the inverse.

In this case, we find $1 = 37*13 - 5*96$

so that 37 is the inverse of 13.

So, to decrypt this message, we use the function:

$$[((x - 32) \times 37) \bmod 96] + 32$$

Thus $Z = 90 \rightarrow 58 \rightarrow 2146 \rightarrow 34 \rightarrow 66 = B$.

So this scheme is not secure, if the algorithm and the key are publicly available.

Letter-by-Letter Encryption

In fact, it is not hard to see that any encryption scheme that encrypts on a letter-by-letter basis cannot be secure.

Can you see why???

If f is the function that encrypts, then regardless of how hard it is to invert ("undo") the function f , one can simply make a table of all encryption values, and then lookup the values in the table.

For example:

value	un-encrypted	encrypted
65	A	M
66	B	Z
67	C	g
68	D	t
69	E	!
70	F	.

Block Encryption

An encryption scheme can be made strong by having it encode large blocks of data all at once, making it infeasible for an attacker to create a table of all blocks and their encrypted values.

Typically, this encryption is done on the binary representation of the data, taking some fixed-size blocks. A typical block size is 256, 512, 1024 or 2048 bits.

For example, with blocks of size 256, if an attacker wished to make a lookup table like the one on the previous slide, the table would have 2^{256} entries, which is about 1.158×10^{77} . Way too big!

We will assume that all of our encryption schemes work in this fashion from now on: They break the data up into blocks of some fixed size, and then encrypt each block separately.

One-Way Functions

Schemes 1 and 2 demonstrate the importance of creating functions that are not easily inverted.

Even for large blocks, the two schemes given are easily inverted, even with a modestly powerful computer.

To make a public key scheme work, we need a function which is easy to compute, but very, very hard to invert.

In this workshop, we will discuss the most widely-used such scheme, RSA. It is based on a problem which has been known to be difficult for over 2500 years: Factoring.

For example, you can “easily” multiply

37975227936943673922808872755445627854565536638199
×40094690950920881030683735292761468389214899724061

But could you factor the product easily?

RSA Challenge Numbers

The two numbers on the previous slide are prime, and their product is the least of the RSA “challenge numbers.”

In 1991, RSA published several numbers which were each the product of two prime numbers. They challenged the world to factor these numbers, offering cash prizes for the winners.

The idea was that if the whole world couldn't claim these prizes (the largest was \$200,000) then their algorithm could be considered secure, and folks should feel safe purchasing and using RSA services.

For some reason, the challenge was rescinded in 2007, with the largest of the prizes unclaimed.

The largest claimed prize was \$20,000 for factoring a 640-digit number. Nothing beyond 663 digits has been factored to this day (June, 2008).

RSA Encryption

Here is the RSA encryption scheme for blocks of size B :

Recipient:

- Select two large primes p and q , and compute their product n , which should be a binary number with at most B bits
- Select a number $e > 1$ which is relatively prime to $(p - 1)(q - 1)$, and compute its inverse $d \bmod (p - 1)(q - 1)$
- Publish n and e . This pair of numbers is the public key. The (secret) decryption key is d .

Sender:

- Break a message into blocks of size B
- To encrypt a message block M , compute $M' = M^e \bmod n$

Recipient:

- To decrypt, compute $(M')^d \bmod n$. This gives back M

Cracking RSA

Do you see how RSA encryption could be cracked if some adversary had the ability to factor quickly?

Since n is public, anyone who could factor n could compute the product $(p - 1)(q - 1)$, and then find the inverse of e under that modulus, yielding the decryption key.

Thus, cracking RSA is no more difficult than factoring.

On the other hand, nobody has proven that factoring n is *required* in order to crack RSA. It may be the case that there is some other way to determine d than by explicitly factoring n . Thus, cracking RSA may not be as hard as factoring.

If it could be shown that finding d from n and e actually enabled one to factor n , then we would know RSA is as hard as factoring. To date, however, this hasn't been done.

RSA Example

Let us select two primes: $p = 13$, $q = 19$.

Their product $n = 247$.

Then $(p - 1)(q - 1) = 216 = 2^3 \times 3^3$

Let $e = 5$, which is relatively prime to 216.

We find $d = 173$, which is $e^{-1} \pmod{216}$.

The recipient publishes the public key (n, e) , which is $(247, 5)$.

In practice, the number “247” would be very, very hard to factor, meaning that the anyone else would not know about the number “216” and so would not be able to compute the inverse of 5.

RSA Example, Continued

The public key is $(247, 5)$

Suppose we wish to send the message “102”.

The sender would compute $102^5 \bmod 247$, which is 163.

The encrypted message “163” would be sent to the receiver, who decrypts it by raising it to the 173 power, mod 247.

And, sure enough, $163^{173} \bmod 247 = 102$.

Large Powers

On the previous slide, we had to compute

$$163^{173} \bmod 247$$

In real applications, these numbers will have, say, 300 or so digits. A clever idea is needed for computing such large powers.

For example, how could a computer compute

$$2^{813278412839741723471823471238} \bmod 247 ?$$

Starting with the number 1, and multiplying by two 813278412839741723471823471238 times is definitely not the way to do it, even if we reduce mod 247 each time.

Instead, we start with 2 and square it repeatedly, and then multiply together some of these squares to obtain our desired power. An example is in order...

Large Powers Example

Let's compute $163^{173} \bmod 247$ using this method.

We start with 163^1 , and square until our exponents can sum to 173, reducing mod 247 each time.

In binary, $173 = 10101101$
so that $163^{173} =$

$$\begin{aligned} &163^{128} \times 163^{32} \times 163^8 \times 163^4 \times 163^1 \\ &= 159 \times 159 \times 159 \times 87 \times 163 \\ &= 102. \end{aligned}$$

163^1	163
163^2	140
163^4	87
163^8	159
163^{16}	87
163^{32}	159
163^{64}	87
163^{128}	159

(Note that after every squaring or multiplication we would reduce the result mod 247, to keep the numbers small.)

This enables us to compute, in under 100 steps, powers like $2^{813278412839741723471823471238} \bmod 247$.

Your Turn with RSA

Suppose we start with the primes 31 and 43. What would the public key be, what would the decryption key be, and what would the message “192” be after encrypted? Assume that for e , you use the smallest possible value.

Your Turn with RSA — Solution

With $p = 31$ and $q = 43$, we have $n = 1333$,
 $(p-1)(q-1) = 30 \times 42 = 1260 = 2^2 \times 3^2 \times 5 \times 7$.

The least e relatively prime to 1260 is 11, which means our public key is (1333, 11).

The inverse of 11 mod 1260 is 1031, so $d = 1031$.

To encrypt 192, we compute $192^{11} \bmod 1333$, which is 491.

To decrypt, we would compute $491^{1031} \bmod 1333$, which gives 192, as expected.

Secret Keys

Secret key cryptography, where the sender and receiver share a key that is used for both encryption and decryption, and is unknown to anyone else, is typically much quicker to perform than a public-key scheme such as RSA.

So in such applications as file transfer (where very large file transfers must be done quickly and securely), handheld devices and embedded applications, it is common to use a shared secret key rather than a public key.

The major difficulty with secret key schemes is getting the same key to the two parties. If one party has the secret key, how can they safely transmit it to the other party?

Key Exchange

There are a few good ways for Alice and Bob to share a secret key. We will discuss three of them:

- Hand deliver the key
- Use a public key to share the secret key
- Diffie-Hellman key exchange

With hand delivery of the key, Alice and Bob can obtain any desired degree of certainty that their secret key has not been observed.

If electronic distribution of the secret key is desired, then the following protocol may be used: Alice publishes a public key, Bob constructs a secret key, and uses Alice's public key to encrypt the secret key and send it to Alice, who alone can decrypt it.

This is typically how online purchases are made. The seller's site publishes a public key, and the buyer uses that key to send a secret key to the seller. This secret key is secure, and faster to use.

Diffie-Hellman Key Exchange

This scheme is interesting in that Alice and Bob exchange a series of messages to obtain some shared key K , but even an eavesdropper who observes all of the messages cannot derive K , even though no public key is used.

- Alice and Bob agree on some large prime p and a number s in the range $2 \dots (p - 1)$
- Alice selects a random number $a < p$, computes $\alpha = s^a \bmod p$, and sends α to Bob.
- Bob selects a random number $b < p$, computes $\beta = s^b$, and sends β to Alice
- Alice computes $\beta^a \bmod p$ and Bob computes $\alpha^b \bmod p$, both of which give the same answer, which becomes their common key K .

An eavesdropper knows only s , p , α and β , from which there is no known feasible way to deduce either a , b or K . This key can then be used in any secret key scheme.

Digital Signatures

In the digital age, it is very easy to alter or forge documents. For example, a picture can be altered, a clause can be added to a Word document or a page deleted from a pdf document.

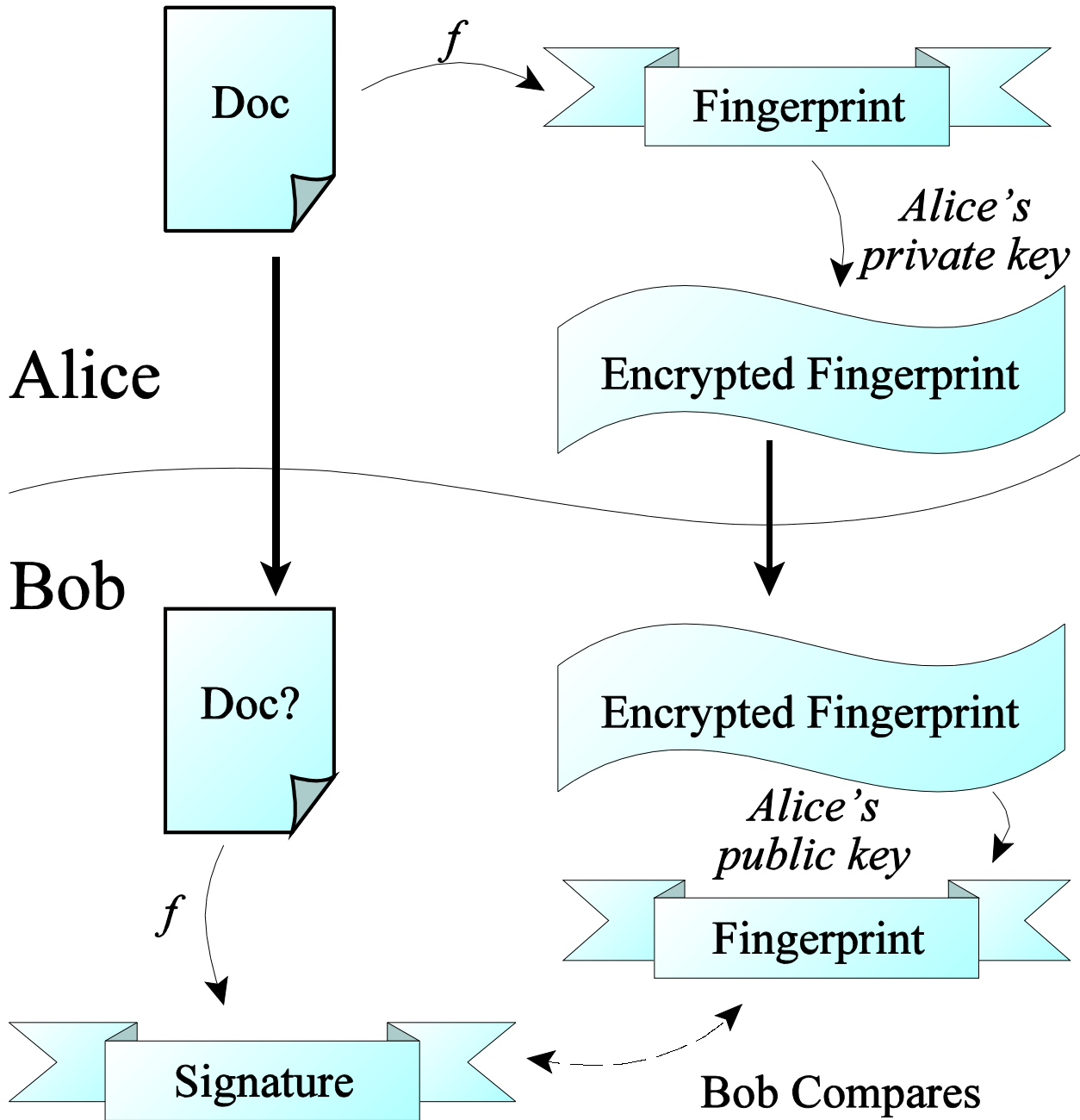
Cryptographic methods give us the ability to protect against this type of attack.

For example, suppose Alice wishes to send a document to Bob, and they both desire some means by which Bob can be certain that the document he receives is an accurate copy of the one Alice sent.

Here is a quick summary of one technique:

- Alice computes a fingerprint of the document, and encrypts it with her private key
- Send the document and the encrypted fingerprint to Bob
- Bob can compute the document fingerprint
- Decrypts the encrypted fingerprint using Alice's public key, and compare with computed one.

Schematic of Digital Signatures



Fingerprints

A fingerprint is the output of any function that takes the document as input. Some fingerprints, of increasing complexity, include:

- Number of characters in the document
- Number of 1's in the binary representation of the document
- List of how many times each character appears in the document
- Break the document into 64-bit blocks, and XOR them all together to obtain a 64-bit signature
- Same as above, but break the 64-bit blocks into eight 8-bit blocks, and permute them according to some predetermined scheme prior to performing the XORs.

Fingerprint Properties

For a fingerprint to be useful, its length should be fixed, regardless of the length of the document that it is supposed to sign. For example, contemporary schemes give on the order of 64 to 256 bit fingerprints and signatures.

Note that fingerprints are examples of what are commonly called *hash functions*.

It should be difficult to modify a document without also modifying its fingerprint. Creation of a false document with the same signature as a known document is called a *preimage attack*.

It should be difficult to create two different documents (of one's own devising) that have the same signature. This avoids the *birthday attack*.

Popular schemes include the Message Digest function MD-5, and the Secure Hash Algorithms in the SHA family.

Handout #1 – ASCII

33	!	49	1	65	A	81	Q	97	a	113	q
34	"	50	2	66	B	82	R	98	b	114	r
35	#	51	3	67	C	83	S	99	c	115	s
36	\$	52	4	68	D	84	T	100	d	116	t
37	%	53	5	69	E	85	U	101	e	117	u
38	&	54	6	70	F	86	V	102	f	118	v
39	'	55	7	71	G	87	W	103	g	119	w
40	(56	8	72	H	88	X	104	h	120	x
41)	57	9	73	I	89	Y	105	i	121	y
42	0	58	:	74	J	90	Z	106	j	122	z
43	0	59	;	75	K	91	[107	k	123	{
44	,	60	<	76	L	92	\	108	l	124	
45	-	61	0	77	M	93]	109	m	125	}
46	.	62	>	78	N	94	^	110	n	126	~
47	/	63	?	79	O	95	_	111	o	127	_
48	0	64	@	80	P	96	`	112	p		

Handout #2 — RSA Encryption

Here is the RSA encryption scheme for blocks of size B :

Recipient:

- Select two large primes p and q , and compute their product n , which should be a binary number with at most B bits
- Select a number $e > 1$ which is relatively prime to $(p - 1)(q - 1)$.
- Compute its inverse $d \pmod{(p - 1)(q - 1)}$
- Publish n and e . This pair of numbers is the public key. The (secret) decryption key is d .

Sender:

- Break a message into blocks of size B
- To encrypt a message block M , compute $M' = M^e \pmod{n}$

Recipient:

- To decrypt, compute $(M')^d \pmod{n}$. This gives back M

Your Turn: Suppose we start with the primes 31 and 43. What would the public key be, what would the decryption key be, and what would the message “192” be after encrypted? Assume that for e , you use the smallest possible value.

Exercises — Public Key Cryptography

Interesting Problems:

1. One weakness of any public key cryptography scheme is that it is easy to decrypt short messages. For example, suppose a government agency wishes to take bids on some contract, and that the bids will be some whole dollar amount under a million dollars. (Typically the bids are placed secretly, so that one company cannot undercut another by bidding one dollar less. Secret bidding encourages all competitors to place their bid as low as possible.) To ensure secrecy, the government agency asks potential contractors to encrypt their bids using the agency's public key using some encryption scheme, perhaps RSA, perhaps not. Show that regardless of the encryption scheme used, anyone with a home computer could discover a competitor's bid from its encrypted value.
2. To overcome the weakness exhibited in the previous problem, public key schemes make use of a technique called *padding*. RSA, for example, might do the following: Suppose that we are using block sizes of 1024 bits when encrypting. When breaking our message into blocks, we will use smaller blocks of size 900 bits (for example) and then append a randomly-generated string of 124 bits to the end of the message prior to encrypting. (This is the *padding* step.) When this message is decrypted, the last 124 bits are simply discarded, and only the first 900 bits are used to reconstruct the original message.

Suppose that Alice wishes to send Bob either a "yes" or a "no," encrypted as described above, and that Eve intercepts the encrypted message. Show that a brute force attempt by Eve to decrypt this message would require her to try at least 21,267,647,932,558,653,966,460,912,964,485,513,216 messages, while without padding she'd only have to try 1.

3. Using the Diffie-Hellman scheme, Alice and Bob have agreed on the prime $p = 11$ and number $s = 2$. Alice has selected $a = 5$, Bob has selected $b = 7$. What would the resulting secret key be, and what parameters would an eavesdropper have been able to discover?
4. Suppose that an eavesdropper listening in on a Diffie-Hellman exchange observes $p = 23$, $s = 3$, $\alpha = 2$ and $\beta = 9$. What are a and b , and what is the shared secret key upon which they would have agreed? Most importantly, how much work was required to find this key, and what would that look like if the prime had had 200 digits instead of 23?
5. Compute, by hand and calculator, the value of $3^{33} \bmod 23$, using the technique of repeated squaring. How many steps would this take if you were computing $3^{333333333} \bmod p$ (on a computer...) mod some prime, p ?
6. Suppose our fingerprinting scheme hashes a bit string as follows: Break the string into Big-blocks of size 16. Within each Big-block, break the Big-block into four small-blocks of four bits, and permute them as follows:
1st, 3rd, 5th, etc... Big-blocks: (ABCD) \rightarrow (BDAC), 2nd, 4th, 6th, etc... Big-blocks: (ABCD) \rightarrow (CDAB).
Finally, take the XOR of the resulting blocks to obtain a 16-bit string. Show that the bit string:
1011 0110 0110 1010 0111 1111 0001 0000 0101 1000 1011 1010 1001 0111 0000 0111
hashes to:
1111 0111 0000 0101.

7. Why is it necessary to encrypt a fingerprint before sending it along with the document?
8. The notion of *message repudiation* is also important in digital signature schemes. Suppose Contractor Chris sends a bid wherein he offers to build a bridge for ten million dollars, and the state accepts his bid and gives him the contract. Later, Chris claims that the contract he sent asked for twelve million dollars, and that the ten-million dollar contract the state is producing must be some sort of forgery. That is, Chris is trying to repudiate the document and claim that he was not its original author. In real life, Chris's signature on the document would make it difficult for him to repudiate the document.
 - a. How can our digital signature notions be used to prevent a document writer from later repudiating that he was the author of the document?
 - b. Analyze your scheme for any potential pitfalls or attack possibilities.