MATH 158
FINAL EXAM
17 DECEMBER 2015

Name : Solutions

- The time limit is three hours.
- No calculators are permitted.
- You are permitted one page of notes, front and back.
- The textbook's summary tables for the systems we have studied are provided at the back.
- For any problem asking you to write a program, you may write in a language of your choice or in pseudocode, as long as your answer is sufficiently specific to tell the runtime of the program.
- Point values are as indicated in the table below.

1	/10	**2**	/10
3	/10	**4**	/10
5	/10	**6**	/10
7	/10	**8**	/10
9	/10	**10**	/10
11	/15	**12**	/15
		$\sum$	/130

(1) Consider the elliptic curve $Y^2 = X^3 + X - 1$ over $\mathbb{Z}/5$.

(a) Determine the number of points on this curve (including the point $\mathcal{O}$).

squares modulo 5: $0^2 \equiv 1$, $1^2 \equiv 1$, $2^2 \equiv 4$, $3^2 \equiv 4$, $4^2 \equiv 1$, ie. 0, 1, and 2.

values of X^3+X-1:

$0^2+0-1 \equiv 4$ ⟹ 2 pts w/ $x=0$
$1^3+1-1 \equiv 1$ ⟹ 2 pts w/ $x=1$
$2^3+2-1 \equiv 4$ ⟹ 2 pts w/ $x=2$
$3^3+3-1 \equiv 4$ ⟹ 2 pts w/ $x=3$
$4^3+4-1 \equiv 2$ ⟹ 0 pts w/ $X=4$.

So there are $2+2+2+2=8$ finite points plus $\mathcal{O}$,

or **9 points** in all.

(b) Determine the order of the point $P = (1, 1)$.

By (a), $\operatorname{ord}(P) \mid 9$, so the order is 1, 3, or 9. It isn't 1 since $P \neq \mathcal{O}$.

We need only check whether or not $3P = \mathcal{O}$.

$2P = (1,1) \oplus (1,1)$

$\lambda = (3\cdot 1^2+1)\cdot(2\cdot 1)^{-1} \equiv 4 \cdot 2^{-1} \equiv 2 \bmod 5$

$x = \lambda^2 - 1 - 1 = 2$

$y = -(1 + 2\cdot(2-1)) \equiv -3 \equiv 2 \bmod 5$

⟹ $2P = (2,2)$

$3P = (2,2) \oplus (1,1)$

~~$\lambda = (2-1)\cdot(2-1)^{-1} \equiv 1 \bmod 5$~~

~~$x = \lambda^2 - 2 - 2 =$~~

Since $(2,2) \neq \ominus(1,1)$, $3P \neq \mathcal{O}$. (no need to compute it).

So $\operatorname{ord} P \neq 1$ or 3; it follows that **$\operatorname{ord} P = 9$**.

(2) Explain briefly why each of the following choices is made in DSA. Be specific about which mathematical facts would make the algorithm either incorrect or insecure otherwise.

(a) The number q is a *prime* number.

q is the order of g, & the sig. scheme is insecure if Eve can do discrete logarithms base g modulo p.

If q is composite, Eve can use Pohlig-Hellman to reduce her effort to instances of DLP where the order of the base is a factor of q; these would be much easier.

q prime ensures this attack is not useful.

(b) The numbers p, q satisfy $p \equiv 1 \pmod{q}$.

$(\mathbb{Z}/p)^\times$ has an element of order q if and only if $q \mid (p-1)$, ie. $p \equiv 1 \bmod q$.

Thus finding g would be impossible & the algorithm wouldn't work otherwise.

(c) The number k is selected *at random*.

If Eve learns k, or can guess, or if Eve finds that a value of k is ever used twice, she can learn the secret signing key from a signature. Choosing a new k at random each time eliminates this risk, making the system more secure.

(3) Alice's RSA public key has modulus N. Bob cannot remember whether her encrypting exponent is 16 or 27. In a well-meaning but very foolish blunder, he decides to encrypt his message m with *both* possible encrypting exponents, creating c_1 (using $e = 16$) and c_2 (using $e = 27$). Bob uses the correct modulus N in both cases. He then sends both c_1 and c_2 to Alice, with an explanation of what happened.

Eve intercepts c_1 and c_2, as well as the information of which exponent was used to create which ciphertext. ~~Show that m can be expressed~~ Express m in terms of c_1 and c_2 using arithmetic modulo N. ~~and hence~~ This shows that Eve can learn the plaintext m.

$$c_1 \equiv m^{16} \bmod N$$

$$c_2 \equiv m^{27} \bmod N$$

$$\Rightarrow c_1^u c_2^v \equiv m^{16u+27v} \bmod N \quad \forall u,v \in \mathbb{Z}.$$

We can use ext. Euclid to express 1 as $16u+27v$:

$$27 = 1 \cdot 27$$

$$16 = 1 \cdot 16$$

$$11 = -1 \cdot 16 + 1 \cdot 27$$

$$5 = 2 \cdot 16 - 1 \cdot 27$$

$$1 = -5 \cdot 16 + 3 \cdot 27$$

So

$$\boxed{m \equiv c_1^{-5} \cdot c_2^{3} \bmod N.}$$

Inverses mod N & powers mod N are both efficient, so Eve can compute m this way very easily.

(4) The following function definition is meant to calculate the sum of two points P, Q on the elliptic curve $Y^2 = X^3 + AX + B$ over $\mathbf{Z}/p$, but it contains a flaw. Explain the case in which the code will not work properly, and how to fix it.

Assumptions: each point (P, Q, or the return value) is either a pair (x, y) of two integers with $0 \leq x, y < p$, or the number 0 (for the point $\mathcal{O}$). You may assume that both P and Q do in fact lie on the curve defined by A and B. Also assume that `inv_mod(a,m)` is a correctly implemented function that returns the inverse of a modulo m whenever a is a unit modulo m, but which results in an error if a is not a unit modulo m.

```
def add(P,Q,A,B,p):
        if P==0: return Q
        if Q==0: return P
        if P[0] == Q[0] and P[1] != Q[1]: return 0 ] *
        if P[0] != Q[0]:
                rise = (P[1] - Q[1]) % p
                run = (P[0] - Q[0]) % p
        else:
                rise = (3*P[0]*P[0] + A) % p
                run = (2*P[1]) % p
        slope = (rise*inv_mod(run,p)) % p
        y_int = (P[1] - P[0]*slope) % p
        x = (slope*slope - P[0]-Q[0]) % p
        y = (-(slope*x + y_int)) % p
        return (x,y)
```

The marked line is meant to detect the case $P = \ominus Q$ and return $\mathcal{O}$ in this case. However, if

$P = (x, 0)$ where $x^3 + Ax + B \equiv 0 \bmod p$
$Q = (x, 0)$

then it will fail; later an error will occur since "run" will be 0 & have no inverse.

An easy fix is to replace this line with:

if P[0]=Q[0] and (P[1]+Q[1])%p ==0: return 0

(5) Write a function `pickg(p,q)` with the following behavior: if p, q are both prime numbers, then the return value must be either a number a between 1 and $p-1$ inclusive with order q modulo p, or the number -1 if no such integer a exists. Your function may be randomized. For full points the (expected value of the) number of arithmetic operations performed by the function must be $\mathcal{O}(\log p)$.

```
import random
def pickg(p,q):
    if (p-1)%q != 0 : return -1
    while True:
        a = random.randrange(2,p)
        g = pow(a, (p-1)/a, p)
        if g != 1: return g
```

(6) Suppose that Samantha is using ECDSA parameters with $q = 7$. She has published two valid signatures: $(2, 3)$ for the document $d = 4$, and $(2, 6)$ for the document $d' = 5$. Eve learns that she used the same random element e to produce both signatures. Determine Samantha's secret signing key, s.

Note. I am withholding the information of Samantha's public key and the system parameters for this problem, since the numbers are small enough that a brute force solution would be possible. In reality, of course, Eve would know all of this, but q would also be large enough that brute force would not be feasible.

$$s_2 \equiv (d + s \cdot s_1) e^{-1} \bmod q$$

i.e. $e \cdot s_2 - s \cdot s_1 \equiv d \bmod q$ for any valid signature.

For these two:

$$e \cdot 3 - s \cdot 2 \equiv 4 \bmod 7$$
$$e \cdot 6 - s \cdot 2 \equiv 5 \bmod 7$$

subtracting the first from the second twice:

$$e \cdot (6 - 2 \cdot 3) + s(-2 + 2 \cdot 2) \equiv 5 - 2 \cdot 4$$

$\iff$ $2s \equiv -3 \equiv 4 \bmod 7$

$\iff$ $\boxed{s \equiv 2 \bmod 7}$

One example of what the full information could be:

curve: $Y^2 = X^3 + 2X + 4$ over $\mathbb{Z}/5$

$G = (4, 1)$ is a pt. of order $q = 7$

secret key $s = 2$ verif. key $V = (2, 4)$ $(= 2 \cdot G)$.

$e = 5$ for both signatures.

(7) Suppose that Eve has intercepted a ciphertext from Bob to Alice. In addition, she knows by other means that the plaintext is one of only 1000 possibilities (for example, it might specify a landmark where Alice and Bob will meet, written in a predictable format and chosen from a short list of options). As usual, Eve knows Alice's public key, but not her private key.

(a) Suppose that the cryptosystem being used is RSA. Explain how Eve can very quickly identify for certain which of the 1000 candidates is the true plaintext.

For each candidate $m_1, m_2, \cdots, m_{1000}$,
Eve just encrypts it, computing
$$c_i \equiv m_i^e \bmod N \quad [(N,e) \text{ is the public key}]$$
Exactly one c_i will be the intercepted ciphertext. Encryption is one-to-one, so Eve knows m_i is the plaintext.

(b) Suppose that the cryptosystem being used is Menezes-Vanstone (table 6.13). Describe a procedure Eve could use that, *with very high probability*, will pick out the correct plaintext from the list. (More formally: your procedure should have the property that if the 999 false plaintexts were chosen uniformly at random, then the probability of choosing one of them should be negligible.)

Let the candidates be (m_{i1}, m_{i2}) for $i = 1, 2, \cdots, 1000$, and the ciphertext be (c_1, c_2).

For each candidate, compute
$$x_i \equiv c_1 \cdot m_{i1}^{-1} \bmod p$$
$$y_i \equiv c_2 \cdot m_{i2}^{-1} \bmod p.$$
If (x_i, y_i) doesn't lie on the curve, the candidate can be thrown out. (if correct, $(x_i, y_i) = (x_T, y_T)$)

The odds of a false candidate not being thrown out are slim, since $\approx \frac{1}{p}$ possible pairs (x, y) actually lie on the curve. So most likely only the true plaintext will remain.

(8) The NTru procedure (table 7.4) stipulates that p and q should be chosen such that $\gcd(p, q) = 1$. Suppose that parameters are chosen that do not obey this rule, and instead $p \mid q$. In this case, the system is completely insecure. Write a function that Eve could use to can break it.

Specifically: write a function `extract(e,N,p,q,d,h)` that efficiently extracts the plaintext **m** from any cipher text **e**, given only the public key and system parameters. The arguments `e` and `h` will be given as lists of N integers. The coefficients in your answer should be either centerlifted modulo p or reduced modulo p in the typical way.

$$\underline{e} \equiv p\,\underline{h} \star \underline{r} + \underline{m} \mod q.$$

Since $p|q$, it follows that all coeffs of both sides are congruent mod p as well as mod q.

So

$$\underline{e} \equiv p\,\underline{h} \star \underline{r} + \underline{m} \mod p$$
$$\equiv 0 \cdot \underline{h} \star \underline{r} + \underline{m} \mod p$$
$$\equiv \underline{m} \mod p.$$

So the function is extremely simple to write.

```
def extract(e, N, p, q, d, h):
    return [ei%p for ei in e]
```

(9) Suppose that P, Q are two points on an elliptic curve over $\mathbf{Z}/9719$ (the number $p = 9719$ is prime). The order of the elliptic curve is a prime number q, and neither P nor Q is $\mathcal{O}$. Alice has constructed the following two lists of points.

$$[\mathcal{O},\ P,\ 2P,\ \cdots,\ 99P]$$
$$[Q,\ Q \ominus 100P,\ Q \ominus 200P,\ \cdots,\ Q \ominus 9900P]$$

Prove that there must exist a common element between these two lists, and describe how finding this common element can be used to find an integer n such that $Q = nP$.

Since q is prime, every non-$\mathcal{O}$ point on the curve has order q. So $\mathcal{O}, P, 2P, \cdots, (q-1)P$ are all distinct (since $iP = jP$ would imply $(i-j)P = \mathcal{O}$, ie $q \mid (i-j)$), hence all pts. on the curve are equal to a multiple of P.

By Hasse's theorem,

$$\begin{aligned} q &\leq p + 1 + 2\sqrt{p} \\ &\leq 9719 + 1 + 2 \cdot \sqrt{9719} \\ &< 9720 + 2 \cdot \sqrt{10000} = 9720 + 200 \\ &< 10.000. \end{aligned}$$

So $\{\mathcal{O}, P, 2P, \cdots, 9999P\}$ includes the point Q somewhere, say at nP ($n < 10.000$). Let

$$\left.\begin{aligned} i &= n \% 100 \\ j &= \lfloor n / 100 \rfloor \end{aligned}\right] \quad \text{so } n = i + 100j$$

Both i, j are between 0 & 99 inclusive. so

iP lies in the first list

& $Q \ominus j \cdot 100P$ lies in the second

& there are equal.

Conversely, if $i \cdot P = Q \ominus (100j)P$ for some i, j, then

$$Q = (i + 100j)P$$

so $n = i + 100j$ is the desired value.

(10) Suppose that the NTru cryposystem (Table 7.4) is modified in the following ways.

- The single integer d in the parameters is replaced with three integers d_1, d_2, d_3 such that $d_1 > d_2 > d_3$. The requirement that $q > (6d+1)p$ is removed.
- When Alice chooses $\mathbf{f}$, she chooses it from $\mathcal{T}(d_1+1, d_1)$.
- When Alice chooses $\mathbf{g}$, she chooses it from $\mathcal{T}(d_2, d_2)$.
- When Bob chooses $\mathbf{r}$, he chooses it from $\mathcal{T}(d_3, d_3)$.

Derive an inequality of the form "$q > \cdots$" (to replace $q > (6d+1)p$ from the original version) in terms of d_1, d_2, d_3 (not all three of which must necessarily be used) and the other public parameters, such that decryption is guaranteed to succeed as long as this inequality holds.

As before, we require that $p\underline{g} \star \underline{r} + \underline{f} \star \underline{m}$ is its own centerlift modulo q in order for decryption to work. That is, all coeff. must be ~~less than~~ in $\left(-\frac{q}{2}, \frac{q}{2}\right]$; it suffices for them to be below $\frac{q}{2}$ in absolute value.

a coeff. of $\underline{g} \star \underline{r}$ can be written as the sum of d_3 coeffs. of $\underline{g}$ minus d_3 coeffs of $\underline{g}$ (since $\underline{r}$ has d_3 $+1$'s & d_3 -1's);

all coeffs of $\underline{g}$ are ± 1 or 0, so $|\underline{g} \star \underline{r}|_\infty \leq 2d_3$, and $|p\underline{g} \star \underline{r}| \leq 2pd_3$.

a coeff. of $\underline{f} \star \underline{m}$ is a sum of d_1+1 coeffs of $\underline{m}$ minus a sum of d_1 terms; all terms of $\underline{m}$ are $\leq \frac{p}{2}$ in abs. value, so $|\underline{f} \star \underline{m}| \leq (2d_1+1)\cdot\frac{p}{2}$.

Thus $|p\underline{g} \star \underline{r} + \underline{m}|_\infty \leq 2pd_3 + (2d_1+1)\cdot\frac{p}{2} = (4d_3 + 2d_1 + 1)\cdot\frac{p}{2}$.

Decryption works as long as this is $< \frac{q}{2}$. Hence it suffices to ensure that

$$\boxed{q > (2d_1 + 4d_3 + 1)p}$$

(note: if $d = d_1 = d_2 = d_3$, we recover the original inequality).

(11) Samantha and Victor agree to the following digital signature scheme. The public parameters and key creation are identical to those of ECDSA. The verification procedure is different: to decide whether (s_1, s_2) is a valid signature for a document d, Victor computes

$$w_1 \equiv s_1^{-1} d \pmod q$$
$$w_2 \equiv s_1^{-1} s_2 \pmod q,$$

then he check to see whether or not

$$x(w_1 G \oplus w_2 V)\% q = s_1.$$

If so, he regards (s_1, s_2) as a valid signature for d.

(a) Describe a signing procedure that Samantha can follow to produce a valid signature on a given document d. The procedure should be randomized in such a way that it will generate different signatures if executed repeatedly on the same document.

Suppose that Samantha takes $s_1 = x(e \cdot G)\% q$ w/ e random as in ECDSA.

The equation $x(w_1 G \oplus w_2 V)\% q = s_1$ of integers is guaranteed by the equation

$$w_1 G \oplus w_2 V = e \cdot G$$

of points on the curve, which is equivalent to the congruence

$$w_1 + w_2 \cdot s \equiv e \mod q$$
$$\iff s_1^{-1} d + s_1^{-1} s_2 s \equiv e \mod q$$
$$\iff d + s \cdot s_2 \equiv e \cdot s_1 \mod q$$
$$\iff s_2 \equiv s^{-1}(e \cdot s_1 - d) \mod q.$$

So Samantha's signing procedure can be:

- Choose a random $e \in \mathbb{Z}/q$.
- $s_1 = x(e \cdot G)\% q$
- $s_2 \equiv s^{-1}(e s_1 - d) \mod q.$

(b) Describe a forgery procedure that Eve can follow to create a signature (s_1, s_2) *and a document* d such that (s_1, s_2) is a valid signature for d under this scheme. Note that Eve does not need to be able to choose d in advance. The procedure should be randomized in such a way that it can generate many different forgeries (on many different documents).

(see P. Set 8 #1.16 for an analogous construction)

As in the examples from class, Eve can get a bit of flexibility by choosing two numbers i, j at random and setting first:

$$s_1 = x(i \cdot G \oplus j \cdot V) \% q$$

(this is as if Samantha chose e to be $i + js$, but Eve can do it without knowing s).

Then she must choose s_2 and d such that:

$$i \cdot G \oplus j \cdot V = w_1 G \oplus w_2 V$$

$$\Leftrightarrow \quad i + ja \equiv s_1^{-1} d + s_1^{-1} s_2 a \mod q$$

$$\Leftrightarrow \quad s_1 i + s_1 j a \equiv d + s_2 a \mod q$$

$$\Leftrightarrow \quad (s_1 i - d) \equiv (s_2 - s_1 j) \cdot a \mod q.$$

Eve can eliminate the need to know a by setting

$$s_2 \equiv s_1 j \mod q$$
$$d \equiv s_1 i \mod q,$$

which will satisfy the desired congruence.

In summary:

1) choose $i, j \in \mathbb{Z}/q$ at random.
2) set (in order):

$$s_1 = x(i \cdot G \oplus j V) \% q$$
$$s_2 \equiv s_1 \cdot j \mod q$$
$$d = s_1 \cdot i \mod q$$

(12) Suppose that n is an odd integer such that exactly $\frac{1}{32}$ of all units modulo n are squares (i.e. are congruent to some integer square modulo n). Alice wishes to factor n. Suppose that Alice chooses m *distinct* elements $a_1, a_2, \cdots, a_m$ of $\{1, 2, \cdots, \frac{n-1}{2}\}$ at random.

(a) Suppose that Alice discovers that $a_i^2 \equiv a_j^2 \pmod{n}$ for some $i \neq j$. Write a function `factor(n,ai,aj)` which returns a proper factor (i.e. a factor besides 1 or n) of n given the values a_i and a_j whose squares are congruent. For full credit, your function should perform no more than $\mathcal{O}(\log n)$ arithmetic operations.

Note that $a_i^2 \equiv a_j^2$ means that $n \mid (a_i+a_j)(a_i-a_j)$. Since $0 < a_i+a_j \leq n-1$, $n \nmid (a_i+a_j)$. Similarly $-n < a_i-a_j < n$ & $a_i-a_j \neq 0$, so $n \nmid (a_i-a_j)$. So For n to divide the product, some factor must divide a_i+a_j, some divide a_i-a_j, but n divides neither. So $\gcd(n, a_i+a_j)$ and $\gcd(n, a_i-a_j)$ are both proper factors. We can find either one with the Euclidean algorithm.

```
def factor(n, ai, aj):      # finds gcd(n, ai+aj).
    a,b = n, ai+aj
    while b ≠ 0:
        a,b = b, a%b
    return a
```

$1-e^{-32\binom{m}{2}/\varphi(n)}$

(b) Assuming that all m of these elements are (distinct) *units* modulo n, prove that the probability that $a_i^2 \equiv a_j^2 \pmod{n}$ for some $i \neq j$ is at least ~~$e^{-32\binom{m}{2}/\phi(n)}$~~. You may assume without proof that $e^{-x} \geq 1 - x$ for all real numbers x.

There are $\varphi(n)/32$ different squares to choose from ($\varphi(n)$ units of which 1/32 are squares). So the prob. of choosing m different values of a_i^2 mod n is:

$$1 \cdot \underbrace{\left(1-\frac{1}{\varphi(n)/32}\right)}_{\text{prob. } a_2^2 \not\equiv a_1^2} \cdot \underbrace{\left(1-\frac{2}{\varphi(n)/32}\right)}_{a_3^2 \not\equiv a_1^2 \text{ or } a_2^2} \cdot \left(1-\frac{3}{\varphi(n)/32}\right) \cdots \underbrace{\left(1-\frac{m-1}{\varphi(n)/32}\right)}_{a_m^2 \not\equiv a_1^2, a_2^2, \ldots, a_{m-1}^2}$$

$$\leq e^0 \cdot e^{-32/\varphi(n)} \cdot e^{-2\cdot 32/\varphi(n)} \cdots e^{-(m-1)\cdot 32/\varphi(n)}$$

$$= e^{-32(0+1+2+\cdots+(m-1))/\varphi(n)} = e^{-32\binom{m}{2}/\varphi(n)}$$

So the prob. that there is <u>some</u> collision is $\geq 1-e^{-32\binom{m}{2}/\varphi(n)}$, as desired.

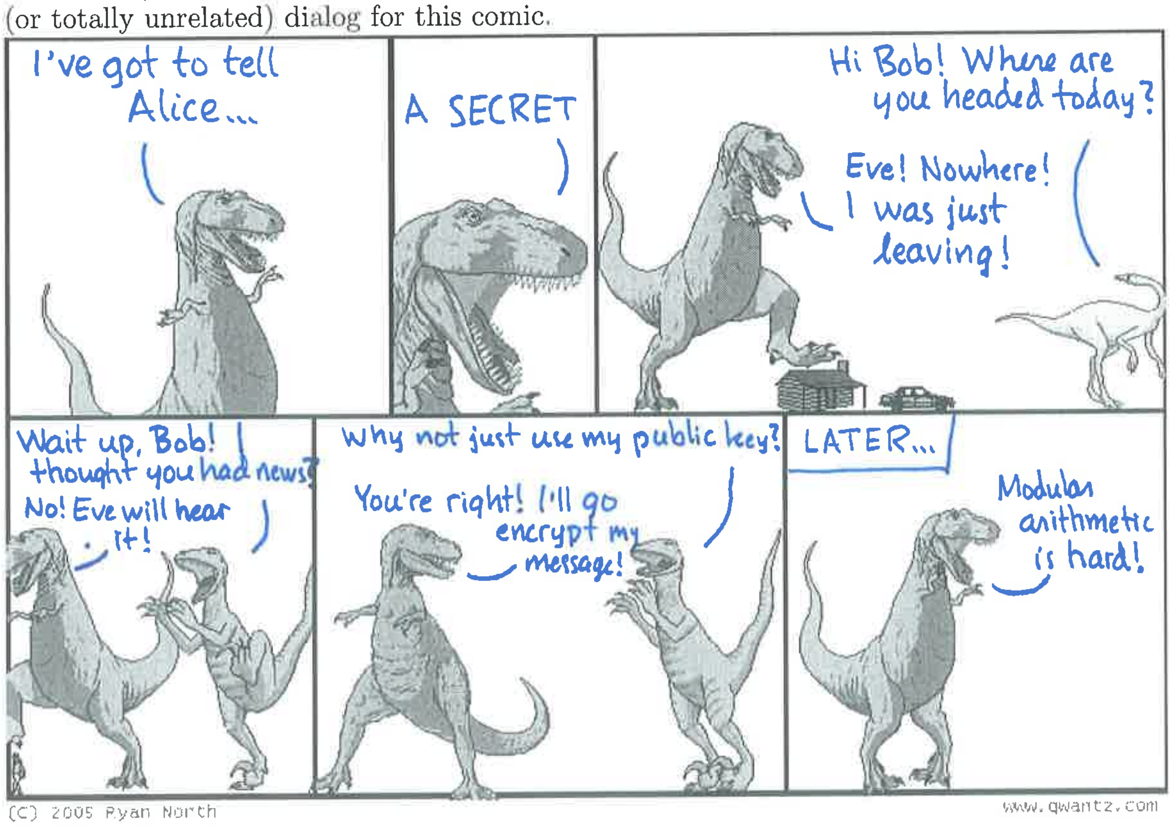
(c) Suppose that the assumption in part (b) fails, and in fact one of the a_i is *not* a unit modulo n. This is a feature, not a bug: describe how Alice can quickly find a proper factor of n in this case, before she even looks for any collisions.

a_i is a nonunit $\iff$ <u>$\gcd(a_i, n) \neq 1$</u>.

Since $a_i < n$, $\gcd(a_i, n)$ is a proper factor of n; we can just return it now and stop looking for more a_i.

(additional space for work)

"**Bonus**" (to keep me happy during grading, not for real points): fill in cryptography-related (or totally unrelated) dialog for this comic.



Public parameter creation	
A trusted party chooses and publishes a (large) prime p and an integer g having large prime order in $\mathbb{F}_p^*$.	
Private computations	
Alice	**Bob**
Choose a secret integer a. Compute $A \equiv g^a \pmod{p}$.	Choose a secret integer b. Compute $B \equiv g^b \pmod{p}$.
Public exchange of values	
Alice sends A to Bob $\longrightarrow$	A
B $\longleftarrow$	Bob sends B to Alice
Further private computations	
Alice	**Bob**
Compute the number $B^a \pmod{p}$.	Compute the number $A^b \pmod{p}$.
The shared secret value is $B^a \equiv (g^b)^a \equiv g^{ab} \equiv (g^a)^b \equiv A^b \pmod{p}$.	

Table 2.2: Diffie–Hellman key exchange

Public parameter creation	
A trusted party chooses and publishes a large prime p and an element g modulo p of large (prime) order.	
Alice	**Bob**
Key creation	
Choose private key $1 \le a \le p-1$. Compute $A = g^a \pmod{p}$. Publish the public key A.	
Encryption	
	Choose plaintext m. Choose random element k. Use Alice's public key A to compute $c_1 = g^k \pmod{p}$ and $c_2 = mA^k \pmod{p}$. Send ciphertext (c_1, c_2) to Alice.
Decryption	
Compute $(c_1^a)^{-1} \cdot c_2 \pmod{p}$. This quantity is equal to m.	

Table 2.3: Elgamal key creation, encryption, and decryption

Bob	Alice
Key creation	
Choose secret primes p and q. Choose encryption exponent e with $\gcd(e, (p-1)(q-1)) = 1$. Publish $N = pq$ and e.	
Encryption	
	Choose plaintext m. Use Bob's public key (N, e) to compute $c \equiv m^e \pmod{N}$. Send ciphertext c to Bob.
Decryption	
Compute d satisfying $ed \equiv 1 \pmod{(p-1)(q-1)}$. Compute $m' \equiv c^d \pmod{N}$. Then m' equals the plaintext m.	

Table 3.1: RSA key creation, encryption, and decryption

Samantha	Victor
Key creation	
Choose secret primes p and q. Choose verification exponent e with $\gcd(e, (p-1)(q-1)) = 1$. Publish $N = pq$ and e.	
Signing	
Compute d satisfying $de \equiv 1 \pmod{(p-1)(q-1)}$. Sign document D by computing $S \equiv D^d \pmod{N}$.	
Verification	
	Compute $S^e \bmod N$ and verify that it is equal to D.

Table 4.1: RSA digital signatures

Public parameter creation	
A trusted party chooses and publishes a large prime p and primitive root g modulo p.	
Samantha	**Victor**
Key creation	
Choose secret signing key $1 \le a \le p-1$. Compute $A = g^a \pmod{p}$. Publish the verification key A.	
Signing	
Choose document $D \bmod p$. Choose random element $1 < k < p$ satisfying $\gcd(k, p-1) = 1$. Compute signature $S_1 \equiv g^k \pmod{p}$ and $S_2 \equiv (D - aS_1)k^{-1} \pmod{p-1}$.	
Verification	
	Compute $A^{S_1} S_1^{S_2} \bmod p$. Verify that it is equal to $g^D \bmod p$.

Table 4.2: The Elgamal digital signature algorithm

Public parameter creation	
A trusted party chooses and publishes large primes p and q satisfying $p \equiv 1 \pmod{q}$ and an element g of order q modulo p.	
Samantha	**Victor**
Key creation	
Choose secret signing key $1 \le a \le q-1$. Compute $A = g^a \pmod{p}$. Publish the verification key A.	
Signing	
Choose document $D \bmod q$. Choose random element $1 < k < q$. Compute signature $S_1 \equiv (g^k \bmod p) \bmod q$ and $S_2 \equiv (D + aS_1)k^{-1} \pmod{q}$.	
Verification	
	Compute $V_1 \equiv DS_2^{-1} \pmod{q}$ and $V_2 \equiv S_1 S_2^{-1} \pmod{q}$. Verify that $(g^{V_1} A^{V_2} \bmod p) \bmod q = S_1$.

Table 4.3: The digital signature algorithm (DSA)

Public parameter creation	
A trusted party chooses and publishes a (large) prime p, an elliptic curve E over $\mathbb{F}_p$, and a point P in $E(\mathbb{F}_p)$.	
Private computations	
Alice	**Bob**
Chooses a secret integer n_A. Computes the point $Q_A = n_A P$.	Chooses a secret integer n_B. Computes the point $Q_B = n_B P$.
Public exchange of values	
Alice sends Q_A to Bob $\longrightarrow$	Q_A
Q_B $\longleftarrow$	Bob sends Q_B to Alice
Further private computations	
Alice	**Bob**
Computes the point $n_A Q_B$.	Computes the point $n_B Q_A$.
The shared secret value is $n_A Q_B = n_A(n_B P) = n_B(n_A P) = n_B Q_A$.	

Table 6.5: Diffie–Hellman key exchange using elliptic curves

Public parameter creation	
A trusted party chooses a finite field $\mathbb{F}_p$, an elliptic curve $E/\mathbb{F}_p$, and a point $G \in E(\mathbb{F}_p)$ of large prime order q.	
Samantha	**Victor**
Key creation	
Choose secret signing key $1 < s < q - 1$. Compute $V = sG \in E(\mathbb{F}_p)$. Publish the verification key V.	
Signing	
Choose document $d \bmod q$. Choose random element $e \bmod q$. Compute $eG \in E(\mathbb{F}_p)$ and then, $s_1 = x(eG) \bmod q$ and $s_2 \equiv (d + ss_1)e^{-1} \pmod{q}$. Publish the signature (s_1, s_2).	
Verification	
	Compute $v_1 \equiv ds_2^{-1} \pmod{q}$ and $v_2 \equiv s_1 s_2^{-1} \pmod{q}$. Compute $v_1 G + v_2 V \in E(\mathbb{F}_p)$ and verify that $x(v_1 G + v_2 V) \bmod q = s_1$.

Table 6.7: The elliptic curve digital signature algorithm (ECDSA)

Public Parameter Creation	
A trusted party chooses and publishes a (large) prime p, an elliptic curve E over $\mathbb{F}_p$, and a point P in $E(\mathbb{F}_p)$.	
Alice	**Bob**
Key Creation	
Chooses a secret multiplier n_A. Computes $Q_A = n_A P$. Publishes the public key Q_A.	
Encryption	
	Chooses plaintext values m_1 and m_2 modulo p. Chooses a random number k. Computes $R = kP$. Computes $S = kQ_A$ and writes it as $S = (x_S, y_S)$. Sets $c_1 \equiv x_S m_1 \pmod{p}$ and $c_2 \equiv y_S m_2 \pmod{p}$. Sends ciphertext (R, c_1, c_2) to Alice.
Decryption	
Computes $T = n_A R$ and writes it as $T = (x_T, y_T)$. Sets $m_1' \equiv x_T^{-1} c_1 \pmod{p}$ and $m_2' \equiv y_T^{-1} c_2 \pmod{p}$. Then $m_1' = m_1$ and $m_2' = m_2$.	

Table 6.13: Menezes–Vanstone variant of Elgamal (Exercises 6.17, 6.18)

Alice	**Bob**
Key Creation	
Choose a large integer modulus q. Choose secret integers f and g with $f < \sqrt{q/2}$, $\sqrt{q/4} < g < \sqrt{q/2}$, and $\gcd(f, qg) = 1$. Compute $h \equiv f^{-1}g \pmod{q}$. Publish the public key (q, h).	
Encryption	
	Choose plaintext m with $m < \sqrt{q/4}$. Use Alice's public key (q, h) to compute $e \equiv rh + m \pmod{q}$. Send ciphertext e to Alice.
Decryption	
Compute $a \equiv fe \pmod{q}$ with $0 < a < q$. Compute $b \equiv f^{-1}a \pmod{g}$ with $0 < b < g$. Then b is the plaintext m.	

Table 7.1: A congruential public key cryptosystem

Public parameter creation	
A trusted party chooses public parameters (N, p, q, d) with N and p prime, $\gcd(p, q) = \gcd(N, q) = 1$, and $q > (6d + 1)p$.	
Alice	**Bob**
Key creation	
Choose private $\boldsymbol{f} \in \mathcal{T}(d+1, d)$ that is invertible in R_q and R_p. Choose private $\boldsymbol{g} \in \mathcal{T}(d, d)$. Compute $\boldsymbol{F}_q$, the inverse of $\boldsymbol{f}$ in R_q. Compute $\boldsymbol{F}_p$, the inverse of $\boldsymbol{f}$ in R_p. Publish the public key $\boldsymbol{h} = \boldsymbol{F}_q \star \boldsymbol{g}$.	
Encryption	
	Choose plaintext $\boldsymbol{m} \in R_p$. Choose a random $\boldsymbol{r} \in \mathcal{T}(d, d)$. Use Alice's public key $\boldsymbol{h}$ to compute $\boldsymbol{e} \equiv p\boldsymbol{r} \star \boldsymbol{h} + \boldsymbol{m} \pmod{q}$. Send ciphertext $\boldsymbol{e}$ to Alice.
Decryption	
Compute $\boldsymbol{f} \star \boldsymbol{e} \equiv p\boldsymbol{g} \star \boldsymbol{r} + \boldsymbol{f} \star \boldsymbol{m} \pmod{q}$. Center-lift to $\boldsymbol{a} \in R$ and compute $\boldsymbol{m} \equiv \boldsymbol{F}_p \star \boldsymbol{a} \pmod{p}$.	

Table 7.4: NTRUEncryt: the NTRU public key cryptosystem