Introduction to Modern Cryptography
Lecture 7

November 26, 2013

Instructor: Benny Chor

Teaching Assistant: Nir Bitansky

School of Computer Science
Tel-Aviv University

Fall Semester, 2013–14, 15:00–18:00
Dan David 201

Course site: http://tau-crypto-f13.wikidot.com/

Lecture 7: Plan

- The Chinese remainder theorem.
- More properties of the RSA public key cryptosystem.
- Quadratic residues and the Jacobi symbol.
- Probabilistic encryption based on QR and QNR over Z_{pq}^*.

Weizmann Distinguished Lectures Day (Dec 10th, 2013)

A three-day celebration of the work of Shafi Goldwasser and Silvio Micali will be held at Weizmann in mid-December 2013, consisting of:
* Dec 10 (TUES): Weizmann Distinguished Lectures Day, featuring talks by Boaz Barak, Irit Dinur, Johan Hastad, and Salil Vadhan, and crowned by lectures of Shafi Goldwasser and Silvio Micali (serving as the annual Amir Pnueli Lectures).

All are welcome, but are asked to notify our administration (by email to distinguished.lectures@weizmann.ac.il) about their intentions to attend.

10-10:45 Boaz Barak (MSR, Cambridge, MA): Games, Proofs, Norms, and Algorithms.
11-11:45 Irit Dinur (Weizmann): Products of Games and Graphs.

(lunch)

13:30-14:15 Johan Hastad (KTH, Stockholm): Approximating Maximum Constraint Satisfaction Problems - A Survey.
14:30-15:15 Salil Vadhan (Harvard): The Complexity of Differential Privacy

(coffee break)

15:45 David Peleg: Introduction for Pnueli Lectures
Oded Goldreich: Introduction to the lecturers
16:00-17:00 Shafi Goldwasser: The Cryptographic Lens
17:15 Richard Karp: Brief remarks about Shafi and Silvio
17:25-18:10 Silvio Micali: Rational Proofs

(reception)

It is a Tuesday, and there will be no class on that day. You are all strongly encouraged to attend this very special and free WIS day (do register though).

The RSA Public Key Cryptosystem (reminder)

- Bob's private information: two large primes p, q.
- Public information: Their product, $m = p \cdot q$. An integer e that is relatively prime to $\phi(m) = (p-1) \cdot (q-1)$.
- More private information: An integer d that is relatively prime to $\phi(m) = (p-1) \cdot (q-1)$ and satisfies $d \cdot e = 1 \mod \phi(m)$.

The RSA Public Key Cryptosystem (reminder)

- Bob's private information: two large primes p, q.
- Public information: Their product, $m = p \cdot q$. An integer e that is relatively prime to $\phi(m) = (p-1) \cdot (q-1)$.
- More private information: An integer d that is relatively prime to $\phi(m) = (p-1) \cdot (q-1)$ and satisfies $d \cdot e = 1 \mod \phi(m)$.

- Messages P are elements in Z_m^*, namely numbers in $[1, \ldots, m-1]$ that are relatively prime to m.
- To encrypt P, compute $C = P^e \pmod m$, and send C to Bob.
- To decrypt C, Bob computes $C^d = P^{d \cdot e} = P \pmod m$.

In fact, encryption/decryption works even in all of Z_m. Thanks to Naor Ahkenazy for pointing this out.

"The above mentioned method should not be confused with the exponentiation technique presented by Diffie and Hellman to solve the key distribution problem"

About Computing the Exponent d from e in RSA

The group Z_{pq}^* ha $\phi(pq) = (p-1)(q-1)$ elements.
Thus for every $x \in Z_{pq}^*$, $x^{(p-1)(q-1)} = 1$, and so ar?thmetic "in the exponent" is modulo $(p-1)(q-1)$.

Let e be relatively prime to $(p-1)(q-1)$. We want to find its inverse modulo $(p-1)(q-1)$, namely a d satisfying
$de = 1 \bmod (p-1)(q-1)$.

The "straightforward way" is to apply the extended gcd algorithm, which produces integers d, a such that $de + a(p-1)(q-1) = 1$.

This is always efficient (linear time in length of pq.)

About Computing the Exponent d from e in RSA, cont.

The alternative is to realize that "in the exponent", we work in the group $Z^*_{(p-1)(q-1)}$, whose order is $\phi((p-1)(q-1))$. The inverse of e can be computed by raising e to the group size minus 1 in $Z^*_{(p-1)(q-1)}$.

That is, $d = e^{\phi((p-1)(q-1))-1} \pmod{(p-1)(q-1)}$.

We note that if the factorization of $(p-1)(q-1)$ is unknown, computing $\phi((p-1)(q-1))$ may be infeasible. Thus, this alternative is viable only if cases where such factorization is easy to produce.

Some RSA Properties

- Deterministic Encryption[1]:
  - Plaintext block P is always encrypted as $P^e \pmod{m}$.
  - So, just like ECB mode encryption of symmetric ciphers, we can detect repetitions.

[1] so cannot be semantically secure, as seen in recitation

Some RSA Properties

- Deterministic Encryption[1]:
  - Plaintext block P is always encrypted as $P^e \pmod m$.
  - So, just like ECB mode encryption of symmetric ciphers, we can detect repetitions.

- RSA is multiplicative (aka multiplication homomorphic):
  - Let P_1, P_2 be two plaintext blocks.
  - Then $E(P_1 \cdot P_2) = E(P_1) \cdot E(P_2)$.
  - This immediately implies that RSA does not behave like a pseudo random function (these are not multiplicative).
  - Multiplicativity implies volnurability to chosen ciphertext attacks.
  - Given a ciphertext $C = P^e \pmod m$, choose at random R and compute $C' = CR^e \pmod m$, which is an encryption of PR, a random looking text.
  - A decryption of $C' = CR^e \pmod m$ will reveal P (after one division mod m).

[1] so cannot be semantically secure, as seen in recitation

The Chinese Remainder Theorem (CRT)

Let $n_1, n_2 > 0$ be two integers such that $\gcd(n_1, n_2) = 1$.
Let $0 \leq a_1 < n_1$ and $0 \leq a_2 < n_2$.

Then there is a unique integer x, $0 \leq x < n_1 \cdot n_2$, such that

$$x = a_1 \pmod{n_1} \text{ and } x = a_2 \pmod{n_2}.$$

The Chinese Remainder Theorem (CRT): Proof

Let $n_1, n_2 > 0$ be two integers such that $\gcd(n_1, n_2) = 1$.
Let $0 \leq a_1 < n_1$ and $0 \leq a_2 < n_2$.

Then there is a unique integer x, $0 \leq x < n_1 \cdot n_2$, such that

$$x = a_1 \pmod{n_1} \text{ and } x = a_2 \pmod{n_2}.$$

Proof: Taking
$x = a_1 \cdot (n_2^{-1} \bmod n_1) \cdot n_2 + a_2 \cdot (n_1^{-1} \bmod n_2) \cdot n_1 \pmod{n_1 n_2}$,
yields the desired claim (we'll verify this on board). ♠

Note that $(n_2^{-1} \bmod n_1)$, $(n_1^{-1} \bmod n_2)$ exist because $\gcd(n_1, n_2) = 1$. They can be computed efficiently either by using extended gcd or by raising to exponents $\phi(n_1) - 1, \phi(n_2) - 1$, correspondingly (this direction requires the factorization of n_1 and n_2).

The Chinese Remainder Theorem (CRT): Sage Examples

```
m=crt(2,8,15,28)
print(m)
print
print(m % 15)
print(m % 28)
```

```
92

2
8
```

```
m=crt(6,33,15,36)
print(m)
print
print(m % 15)
print(m % 36)
```

```
141

6
33
```

Sage does not have a built in supports for CRT with more than two moduli. But yes, we can extend it ourselves!

CRT for Three

Let $n_1, n_2, n_3 > 0$ be two integers that are pairwise relatively ptime (for $i \neq j$, $\gcd(n_i, n_j) = 1$).
Let $0 \leq a_1 < n_1$, $0 \leq a_2 < n_2$, and $0 \leq a_3 < n_3$.

Then there is a unique integer x, $0 \leq x < n_1 \cdot n_2$, such that

$$x = a_1 \pmod{n_1},\ x = a_2 \pmod{n_2} \text{ and } x = a_3 \pmod{n_3}.$$

Proof: Computing

$$\begin{aligned} x = & a_1 \cdot ((n_2 n_3)^{-1} \bmod n_1) \cdot n_2 n_3 \\ & + a_2 \cdot ((n_1 n_3)^{-1} \bmod n_2) \cdot n_1 n_3 \\ & + a_3 \cdot ((n_1 n_2)^{-1} \bmod n_3) \cdot n_1 n_2 \end{aligned}$$

and then taking $x \pmod{n_1 n_2 n_3}$ yields the desired number. ♣

CRT for Three, in Sage

We first need to compute inverses, $n^{-1} \pmod{m}$.

```
def inverse(n,m):
    """ inverse of n modulo m """
    if gcd(n,m)!=1:
        return None
    else:
        return pow(n,euler_phi(m)-1,m).lift()
        # euler_phi(m) is a built in Sage function
        # lift() coerces mod m to integers. Just a nuisance.
```

Let's run an example or two

```
inverse(3,7)
```

```
5
```

```
inverse(34,195)
```

```
109
```

CRT for Three, in Sage, cont.

```
def crt3(a1,a2,a3,n1,n2,n3):
    if gcd(n1,n2)!=1 or gcd(n1,n3)!=1 or gcd(n2,n3)!=1:
        return None
    else:
        return (a1*inverse(n2*n3,n1)*n2*n3\
        +a2*inverse(n1*n3,n2)*n1*n3\
        +a3*inverse(n1*n2,n3)*n1*n2)\
        %(n1*n2*n3)
```

Let's run an example or two

```
m=crt3(1,2,3,4,5,7)
print(m)
print
print(m % 4)
print(m % 5)
print(m % 7)
```

```
17

1
2
3
```

Using CRT to Think About Z_{pq}^*

Given $x \in Z_{pq}^*$, it is fruitful to think about its remainders modulo p and q, namely $y = x \pmod{p}$ and $z = x \pmod{q}$.

By the CRT, y and z determine x (and vice versa).

Furthermore, it is often more convenient (and sometimes more efficient) to work with y and z instead of x.

However, this "privilege" is only available to Bob (who knows p and q), not to Eve.

Using CRT to Speed Up RSA Decryption

Goal: Given the encrypted message, C, compute $C^d \pmod{pq}$.
Suffices by CRT to get $C^d \pmod{p}$ and $C^d \pmod{q}$.

- CRT preprocessing (done once): Given p, q, compute $p^{-1} \pmod{q}$ and $q^{-1} \pmod{p}$.
- Decryption, on input C:
  - Bob (having learnt about CRT) wants to computes $y_1 = (C \bmod p)^d \pmod{p}$ and $y_2 = (C \bmod q)^d \pmod{q}$.
  - Bob (having learnt about arithmetic in Z_p^*, Z_q^*) further realizes it suffices to work with smaller exponents $y_1 = (C \bmod p)^{d \pmod{p-1}} \pmod{p}$ and $y_2 = (C \bmod q)^{d \pmod{q-1}} \pmod{q}$.
  - Finally, Bob computes $P = q \cdot (q^{-1} \bmod p) \cdot y_1 + p \cdot (p^{-1} \bmod y_2 \pmod{m}$.
- This yields a non negligible saving (factor 4) in decryption runtime – essentially we deal with two half size exponentiations.

Using a Small Exponent to Speed Up RSA Encryption

- The public exponent, e, can be small
  - This will substantially improve efficiency of encryption.
  - It does not mean that the private exponent, d, is small too.
  - Neither does it seem to make it easier to derive d.
  - Small public exponents are used in practice.
  - Most popular values are either 3 or $2^{16} + 1$.
  - The public exponent, e and $\phi(m) = (p-1)(q-1)$ should still be relatively prime.
  - For $e = 3$ this means 3 should divide neither $p - 1$ nor $q - 1$.

Using a Small Exponent to Speed Up RSA Encryption (2)

- Small exponents are fine for sending a single message to a single user, Bob.
- But they are vulnerable if the same message is to be sent to multiple users.
  - For example, suppose Bob, Bobi, and Bubchek are all using $e = 3$ (with public moduli m_1, m_2, m_3, respectively).
  - Alice wishes to send the plaintext message P to each of them.
  - She encrypts the plaintext message P and sends $P^3 \pmod{m_1}$, $P^3 \pmod{m_2}$, $P^3 \pmod{m_3}$.
  - Notice that $P < m_1, m_2, m_3$, and thus $P^3 < m_1 \cdot m_2 \cdot m_3$.
  - Denote $x = P^3$ (as an integer, not modulo anything).
  - Eve heard $x \pmod{m_1}$, $x \pmod{m_2}$, $x \pmod{m_3}$. She also knows that $x < m_1 \cdot m_2 \cdot m_3$.

Using a Small Exponent to Speed Up RSA Encryption (3)

- Eve heard $x \pmod{m_1}$, $x \pmod{m_2}$, $x \pmod{m_3}$. She also knows that $x < m_1 \cdot m_2 \cdot m_3$.
- m_1, m_2, m_3 are pairwise relatively prime.
- Eve (who overheard our Friday class on CRT and followed every iota) knows that this enables her to efficiently compute $y = x \pmod{m_1 m_2 m_3}$.
- But since $x < m_1 \cdot m_2 \cdot m_3$, this implies that $y = x$ (over the integers!). Therefore Eve now holds $P^3 = x$ without any modular subtractions.
- Computing cubic roots over the integers is easy.
- So Eve finds P and successfully retrieves the plaintext.

Using a Small Exponent to Speed Up RSA Encryption (3)

- Eve heard $x \pmod{m_1}$, $x \pmod{m_2}$, $x \pmod{m_3}$. She also knows that $x < m_1 \cdot m_2 \cdot m_3$.
- m_1, m_2, m_3 are pairwise relatively prime.
- Eve (who overheard our Friday class on CRT and followed every iota) knows that this enables her to efficiently compute $y = x \pmod{m_1 m_2 m_3}$.
- But since $x < m_1 \cdot m_2 \cdot m_3$, this implies that $y = x$ (over the integers!). Therefore Eve now holds $P^3 = x$ without any modular subtractions.
- Computing cubic roots over the integers is easy.
- So Eve finds P and successfully retrieves the plaintext.

- Is this a weakness of small exponent RSA? No!
- It is a weakness in the manner small exponent RSA was used.
- In general, if we have cryptographic building blocks that are secure on their own, this does not mean we can compose or interleave them any way we want, and still maintain security.

Random Self Reducibility of RSA (reminder)

- Let $m = pq, e$ be a given RSA modulus, of length n bits, plus a RSA exponent.
- Suppose $\mathcal{A}$ is an deterministic n^c time algorithm which, on input $E(x) = x^e \pmod m$, outputs x for εm of the inputs in Z_m.

Random Self Reducibility of RSA (reminder)

- Let $m = pq, e$ be a given RSA modulus, of length n bits, plus a RSA exponent.
- Suppose $\mathcal{A}$ is an deterministic n^c time algorithm which, on input $E(x) = x^e \pmod m$, outputs x for εm of the inputs in Z_m.
- Then $\mathcal{A}$ can be converted into a randomized $\varepsilon^{-1} n^c$ expected time algorithm, $\mathcal{R}$ which, on input $E(x) = x^e \pmod m$, outputs x for all the inputs in Z_m.
- Whenever $\mathcal{R}$ terminates, it gives the correct output x.
- Proof on board (using the modular circle).

Random Self Reducibility of RSA (reminder)

- Let $m = pq, e$ be a given RSA modulus, of length n bits, plus a RSA exponent.
- Suppose $\mathcal{A}$ is an deterministic n^c time algorithm which, on input $E(x) = x^e \pmod m$, outputs x for εm of the inputs in Z_m.
- Then $\mathcal{A}$ can be converted into a randomized $\varepsilon^{-1} n^c$ expected time algorithm, $\mathcal{R}$ which, on input $E(x) = x^e \pmod m$, outputs x for all the inputs in Z_m.
- Whenever $\mathcal{R}$ terminates, it gives the correct output x.

- Proof on board (using the modular circle).
- Consequence: For a given $m = pq$ and e, inverting RSA is either hard everywhere or easy everywhere.

Towards Semantic Security: Random Padding ("Salting")

- Padding the message by a block of random bits: Suppose the length of pq is n bits. Use ℓ bits for the message P, concatenate with $n - \ell$ random bits string, r: $E(r \circ P) = (r \circ P)^e \pmod{pq}$.
- Padding reduces the information rate, but increases security. It can be shown that if $n - \ell$ is very large, then padded RSA is resistant to chosen plaintext attack (under appropriate assumption on hardness of inverting the RSA function).
- For security to hold, pad must be random. Choosing $r = \texttt{hello world}$, or any other fixed text, is not a good practice.
- For protection against chosen ciphertext attack, a combination of fixed and random padding was proposed by RSA labs: Let P be a ℓ bit long message. Pad and encrypt by $(00000000 \circ 00000010 \circ r \circ 00000000 \circ P)^e \pmod{pq}$.
- Fixed parts of pad intended to foil multiplication attacks.
- Unfortunately, some chosen ciphertext attacks were later found. Still, scheme is being used.

Semantic Security - Informal Reminder

Cannot distinguish between encryptions of different messages M_1, M_2 (assume equal length).

Suffices to deal with single bit messages $M_1 = 0, M_2 = 1$.

General Remark on Public Key Cryptosystems

- PKCs are order of magnitude slower than private key systems. Hence used mainly to exchange keys or signing.
- Under suitable complexity assumptions, PKC are secure, provided we can trust the association of keys with users.
- If I were tricked to send a message using what I think is the public key of Esau, but Jacob (a well known trickster) is the one that can decipher it, then I may be in trouble.

Isaac rejecting Esau, by Giotto di Bondone, 13-14th centuries, Assisi, Italy.

General Remark on Public Key Cryptosystems

- PKCs are order of magnitude slower than private key systems. Hence used mainly to exchange keys or signing.
- Under suitable complexity assumptions, PKC are secure, provided we can trust the association of keys with users.
- If I were tricked to send a message using what I think is the public key of Esau, but Jacob (a well known trickster) is the one that can decipher it, then I may be in trouble.

Isaac rejecting Esau, by Giotto di Bondone, 13-14th centuries, Assisi, Italy.

- To achieve secure communication without prior physical contact, have to establish (and trust) centers for distributing certificates.
- Should discussed this under "public key infrastructure", but we do not have enough time :-(

Real World Usage of RSA

(1) Key exchange.

Key subsequently used in a symmetric encryption scheme (which typically is much faster).

(2) Digital signatures.

Signatures will be discussed in 1-2 weeks.

Probabilistic Encryption

Goldwasser and Micali, 1982.

Probabilistic Encryption: Goal

Whatever is efficiently computable about the cleartext given the ciphertext, is also efficiently computable without the ciphertext.

Probabilistic Encryption: Goal

Whatever is efficiently computable about the cleartext given the ciphertext, is also efficiently computable without the ciphertext.

In the context of public key encryption, this noble goal will boil down to a rather simple requirement – indistinguishability of encryptions of 0 from encryptions of 1.

There is much more to this paradigm. And rest assured – it will be explored (when we discuss zero knowledge proofs).

QRs and NQRs in Z_{pq}^*

- By now, you all know about quadratic residues in Z_p^*.
- Some of you may even understand quadratic residues in Z_q^*.

- We want to build intuition about quadratic residues and quadratic non residues in Z_{pq}^*.
- We will throw in the Jacobi symbol, $(\frac{x}{pq})$ as well.
- But first, lets examine square roots of 1 in Z_{pq}^*.

Square Roots in Z_{pq}^*

Start with square roots of 1.

- in Z_p^*, 1 has two square roots: 1 and $p-1$.
- in Z_q^*, 1 has two square roots: 1 and $q-1$.
- What about the square roots of 1 in Z_{pq}^*?

- $y^2 = 1 \pmod{pq}$ iff $y^2 = 1 \pmod{p}$ and $y^2 = 1 \pmod{q}$.
- So $y = \pm 1 \pmod{p}$ and $y = \pm 1 \pmod{q}$.
- This gives rise to four systems of modular equations
  1. $y = 1 \pmod{p}$ and $y = 1 \pmod{q}$.
  2. $y = -1 \pmod{p}$ and $y = -1 \pmod{q}$.
  3. $y = 1 \pmod{p}$ and $y = -1 \pmod{q}$.
  4. $y = -1 \pmod{p}$ and $y = 1 \pmod{q}$.
- The solution to (1) is $y = 1$.
- The solution to (2) is $y = pq - 1 = -1 \pmod{p}$.
- The solutions to (3) and (4) are obtained using the Chinese remainder theorem. See a concrete example in the next slide.

Square Roots in Z_{pq}^*: Sage Code and Execusion

```
def unitsqrt(p,q):
    m=p*q
    print(m)
    print
    root1=crt(1,1,p,q)
    root2=crt(-1,-1,p,q)
    root3=crt(1,-1,p,q)
    root4=crt(-1,1,p,q)
    print(root1,root1^2 % m)
    print(root2,root2^2 % m)
    print(root3,root3^2 % m)
    print(root4,root4^2 % m)
    return(root1,root2,root3,root4)
```

Let's run an example or two

```
unitsqrt(13,23)
```

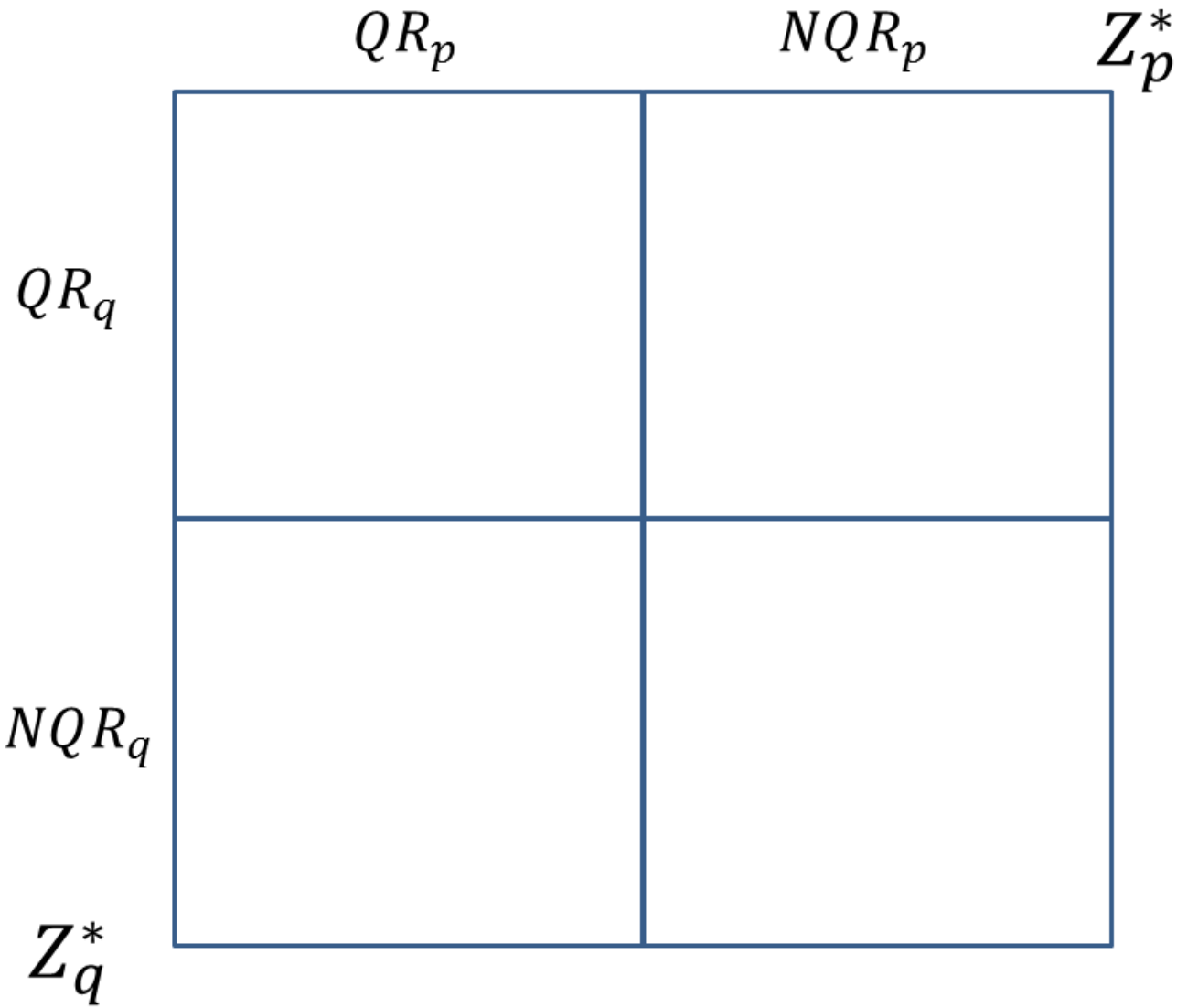
```
299

(1, 1)
(298, 1)
(183, 1)
(116, 1)
(1, 298, 183, 116)
```

Square Roots in Z_{pq}^*

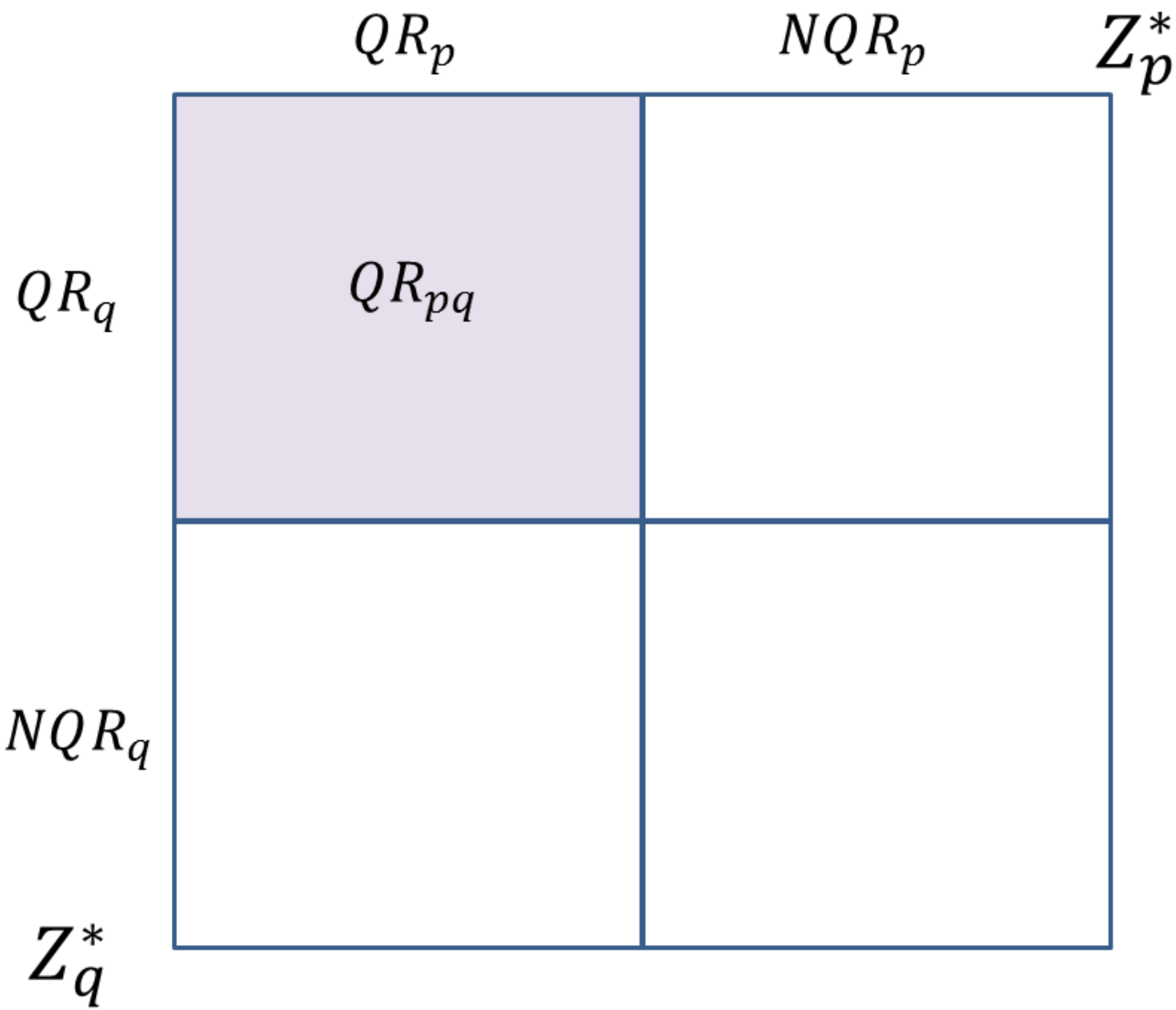
In general, the square roots of y^2 are any of the four square roots of $1 \pmod{pq}$, multiplied by y.

- This implies that every square in Z_{pq}^* has exactly four square roots.
- The number of quadratic residues in Z_{pq}^* is $(p-1)(q-1)/4$.
- The mapping $x \rightarrow x^2 \mod pq$ is a four to one mapping.

The Magic Square(s) in Z^*_{pq}, take 1



The Magic Square(s) in Z^*_{pq}, take 2



Identifying Quadratic Residues and Non Residues in Z_{pq}^*

Given $x \in Z_p^*$, we know how to efficiently test if x is a QR or NQR.
$x \in Z_p^*$ iff $x^{(p-1)/2} = 1 \pmod{p}$.
The value $x^{(p-1)/2}$ is called the Legendre symbol of x modulo p, and denoted $(\frac{x}{p})$.

What about QR or NQR for $x \in Z_{pq}^*$?

Well, if we knew p (and q), this would be a piece of cake.

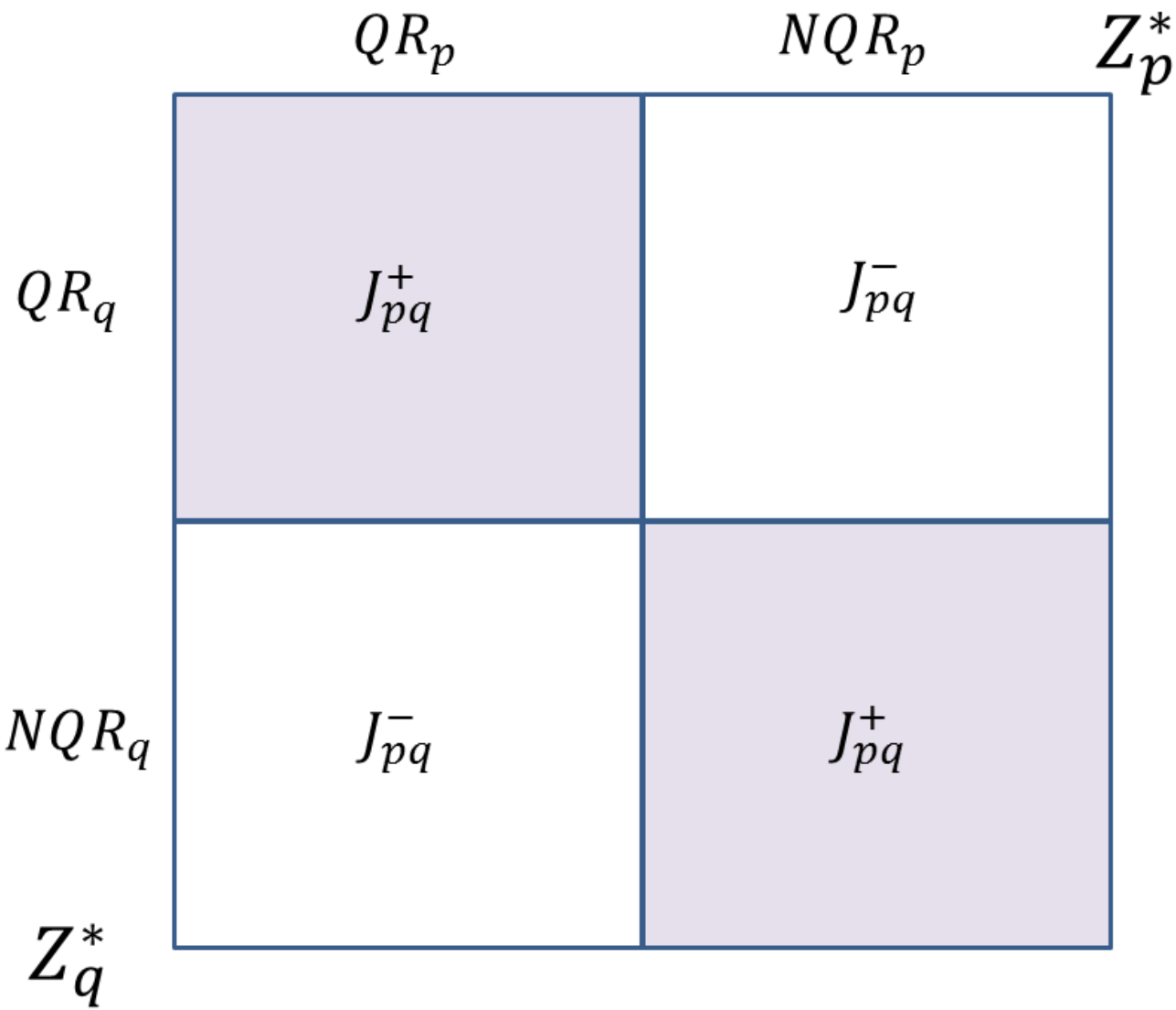
But life is tough, and factoring integers is hard. What can be said if pq is given, but not its factorization?

We cannot tell QR from QNR, but do know something (refer to magic square).

The Jacobi Symbol in Z^*_{pq}

- Let $m = pq$ and $z \in Z^*_{pq}$.
- The Jacobi symbol $(\frac{z}{pq})$ is 1 if $z \bmod p$ and $z \bmod q$ are both quadratic non residues in the appropriate field, or both are quadratic residues. Otherwise, it is -1.
- Interestingly enough, $(\frac{z}{pq})$ can be efficiently computed even without knowing the factorization of $m = pq$.
- The algorithm uses the so called law of quadratic reciprocity, which leads to an algorithm similar to Euclid's gcd algorithm.
- We will skip the proof's details (but you can look it up, *e.g.* in `http://en.wikipedia.org/wiki/Jacobi_symbol`).

The Magic Square(s) in Z^*_{pq}, Refined



Generating Quadratic Residues and Non Residues in Z^*_{pq}

- It is easy to generate a uniformly distributed quadratic non residues in $x \in Z^*_p$.
- It is also easy to generate a uniformly distributed quadratic residues in $x \in Z^*_p$.
- The same applies to $y \in Z^*_q$.
- Knowing the factorization p, q of pq, it is easy to generate a uniformly distributed $z \in Z^*_{pq}$ which is any of the four types modulo p, q:
  (QR,QR), (QR,QNR), (NQR,QR), (QNR,QNR).

Quadratic Non Residues in Z_{pq}^*, cont.

- Knowing the factorization p, q of pq, it is easy to generate a uniformly distributed $z \in Z_{pq}^*$ which is a quadratic non residue modulo p and modulo q.
- How?
- We generate at random a quadratic non residue modulo p, denoted by $x \in Z_p^*$.
- We generate at random a quadratic non residue modulo q, denoted by $y \in Z_q^*$.
- Using the Chinese remainder theorem (CRT), we efficiently find a $z \in Z_{pq}^*$ such that $z = x \pmod{p}$ and $z = y \pmod{q}$.

Quadratic Non Residues in Z^*_{pq}: Special Case

- Claim: Suppose $p = 3 \bmod 4$. Then -1 is a QNR modulo p.
- Proof: x is a QNR modulo p iff $x^{(p-1)/2} = -1 \bmod p$.
- There is an integer k such that $p = 4k + 3$. Thus $(-1)^{(p-1)/2} = (-1)^{((4k+3)-1)/2} = (-1)^{2k+1} = -1$.
- Therefore if $p = 3 \bmod 4$ and $q = 3 \bmod 4$, then -1 is a QNR modulo p and modulo q.

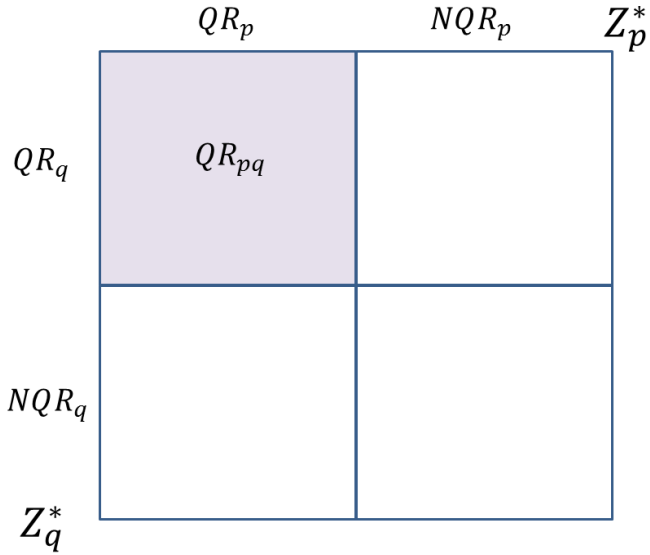
- Incidentally, a product pq where $p = 3 \bmod 4$, $q = 3 \bmod 4$ is called a (Manuel) Blum integer.

The Quadratic Residuosity Assumption (QRA) in Z^*_{pq}

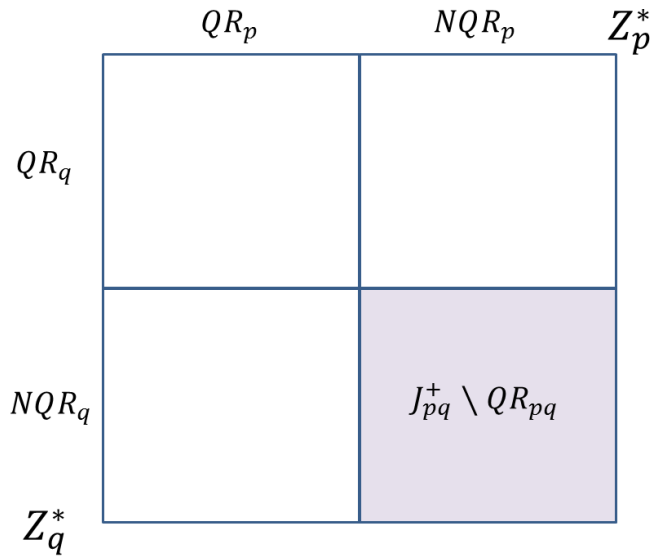
- Let $m = pq$ but not its factorization and a $z \in Z^*_{pq}$ such that $z \bmod p$ and $z \bmod q$ are both quadratic non residues in the appropriate field be given.
- Consider the distributions (think of action on magic square)
  $QR = \{y^2 \bmod pq \mid y \in_R Z^*_{pq}\}$,
  $QNR = \{zy^2 \bmod pq \mid y \in_R Z^*_{pq}\}$,
  where $y \in_R Z^*_{pq}$ is chosen uniformly at random.

- The QRA: QR and QNR are polynomially indistinguishable.

QRA in Z^*_{pq}, Viewed with the Magic Square(s)



is indistinguishable from



A QRA Based Public Key Probabilistic Encryption Scheme

- Bob's private information: two large primes p, q.
- Public information: Their product, $m = p \cdot q$. An integer $z \in Z^*_{pq}$, which is a quadratic non residue modulo p and q.

[2] if one is a square and the other is not, Bob concludes that Alice deviated from the protocol.

A QRA Based Public Key Probabilistic Encryption Scheme

- Bob's private information: two large primes p, q.
- Public information: Their product, $m = p \cdot q$. An integer $z \in Z^*_{pq}$, which is a quadratic non residue modulo p and q.

- Messages P are simply binary strings, $P = b_0 b_1 b_2 \dots b_k$.
- To encrypt b_i, Alice chooses $y_i \in Z^*_{pq}$ at random. If $b_i = 0$, its encryption is $c_i = y^2 \bmod pq$. If $b_i = 1$, its encryption is $c_i = zy^2 \bmod pq$.
- Alice sends $c_0 c_1 c_2 \dots c_k$ to Bob.

- To decrypt $c_0 c_1 c_2 \dots c_k$, Bob computes, for every i, $c_i \pmod p$ and $c_i \pmod q$. If both are squares, Bob determines that $b_i = 0$, otherwise if both are non squares, he determines that $b_i = 1$ [2].

[2] if one is a square and the other is not, Bob concludes that Alice deviated from the protocol.

A QRA Probabilistic Encryption Scheme: Q&A

What does Bob know that Eve does not?

Bob holds p and p. This enables him to determine if c_i is a QR modulo p and p or QNR modulo p and p.

By the quadratic residuosity assumption, Eve cannot determine this information on her own.

A QRA Probabilistic Encryption Scheme: Q&A

Does the fact that Bob published a QNR z not helping Eve?

No! If Eve picks a $z \in Z_{pq}^*$ at random, it will be a QNR modulo p and p with probability exactly $1/4$.

Furthermore, for $p = 3 \bmod 4$ and $q = 3 \bmod 4$, Bob could use $z = -1 (= pq - 1)$ and Eve could do this as well, without weakening QRA.

A QRA Probabilistic Encryption Scheme: Q&A

Is the scheme not very wasteful in terms of communication bandwidth?

This is hard to deny.

But, as we will soon see, this construction buys us provable immunity to chosen plaintext attacks (under QRA, of course).

In addition, this construction was the first one of its kind, and paved the way to more efficient ones, and in fact to the modern complexity based approach to cryptography.

Multiplicative Homomorphism of QR/QNR in Z_{pq}^*

We are given $m = p \cdot q$ and an integer $z \in Z_{pq}^*$, which is a quadratic non residue modulo p and q.

Let $b_1, b_2 \in \{0, 1\}$, and denote $E(0) =$QR, $E(1) =$QNR (where QR, QNR denote a random quadratic residue/non residue modulo pq.

Since QR·QR=QR, QNR·QNR=QR,
QNR·QR=QNR, and QR·QNR=QNR,
we have

$E(b_1) \cdot E(b_2) = E(b_1 \oplus b_2)$.

Semantic Security (Sketch)

First, we observe that due to the system being a public key one, Eve could generate herself encryptions of any (polynomially many) strings she chooses, $s_1, \ldots, s_\ell$.

Suppose there is a polynomial time computable non constant predicate $C(b_1 \ldots b_k)$, and an adversary algorithm $\mathcal{A}$ that on encrypted input $E(b_1) \ldots E(b_k)$ has an advantage (over the background probability) of computing $C(b_1 \ldots b_k)$.

Using a so called hybrid argument, it can be shown that the advantage in computing $C(b_1 \ldots b_k)$ can be turned into an advantage in computing b_i from $E(b_i)$ for some bit b_i.

This means Eve can efficiently distinguish QRs from RNRs.

This contradicts the quadratic residuosity assumption. ♠

Self Reducibility of QR/QNR in Z^*_{pq}

We are given $m = p \cdot q$ and an integer $z \in Z^*_{pq}$, which is a quadratic non residue modulo p and q. Let n be the length of m (in bits).

Suppose $\mathcal{A}$ is a polynomial time (polynomial in n) algorithm that on input $x \in Z^*_{pq}$ whose Jacobi symbol is 1, outputs a "guess" to x being a QR or QNR. Furthermore, suppose this "guess" is correct with probability $\frac{1}{2} + \varepsilon$ (probability is over choices of x and coin tosses of $\mathcal{A}$).

Claim: Given $\mathcal{A}$ as above, there is an algorithm $\mathcal{B}$ that runs in time polynomial in $(n + \varepsilon^{-1})$, such that on input $t \in Z^*_{pq}$ whose Jacobi symbol is 1, outputs if t is a QR or QNR modulo pq, and furthermore, $\mathcal{B}$ succeeds with probability $\geq 1 - 2^{-n}$.

Self Reducibility of QR/QNR in Z^*_{pq}: Proof Idea

How do we turn a very slight advantage (probability $\frac{1}{2} + \varepsilon$ of guessing x being a QR or QNR), to an overwhelming probability of being correct (probability $\frac{1}{2} + \varepsilon$ of guessing x being a QR or QNR)?

1. "Spread" x over all elements in Z^*_{pq} with Jacobi symbol 1. Do this by computing xy_i^2 and zxy_j^2 for polynomially many random independent y_i and y_j.
2. Query $\mathcal{A}$ for QR/QNR of all these xy_i^2 and zxy_j^2.
3. Count how many answers indicate that x is a QR, and how many that it is a QNR.
4. Take the majority. Show probability of error is small $\leq 2^{-n}$ by using a specific inequality related to laws of large numbers (actually large deviations), *e.g.* the Chernoff bound.

Comment: This sampling and taking majority should be familiar to those of you attempting to solve the bonus problem (number 7) in HW2.

Weak vs. Strong QRA

- Strong QRA: QR and QNR are polynomially indistinguishable.
- Weak QRA: Any efficient algorithm trying to distinguish QR from QNR errs with probability at least $1/2^n$ (n being the length of pq).

Self Reducibility of QR/QNR in Z^*_{pq} implies weak and strong QRA are equivalent.

Signatures

Signatures

http://lacourphoto.net/uploaded_images/signatures1-770492.jpg

Hand Written Signatures

- Relate an individual, through a handwritten signature, to a document.
- Signature can be verified against a prior authenticated one, which was signed in person in a bank, in the presence of a public notary public, etc.
- Should be hard to forge.
- Are legally binding (convince a third party, e.g. a judge).

Digital Signature Schemes

- Relate an individual, through a digital string, to a document.
- Would like to achieve all features of hand written signatures, plus more.
- For example, should be able to base difficulty of forgery on some hard computational problem, not just on ineptitude of forger.
- Diffie and Hellman were first to propose such framework.
- An implementation was first given by RSA.

Diffie and Hellman "New Directions in Cryptography" (76)

Diffie and Hellman proposed a "textbook framework", based on any deterministic public key cryptosystem.

We will describe this framework and implementation(s), and then discuss some specific shortcomings of it.

- Let E_A be Alice's public encryption key, and let D_A be Alice's private decryption key.
- To sign the message M, Alice computes the string $y = D_A(M)$ and sends (M, y) to Bob.
- To verify this is indeed Alice's signature, Bob computes the string $x = E_A(y)$ and checks that indeed $x = M$.

Diffie and Hellman "New Directions in Cryptography" (76)

Diffie and Hellman proposed a "textbook framework", based on any deterministic public key cryptosystem.

We will describe this framework and some implementation(s), and then discuss some specific shortcomings of it.

Intuition: Only Alice can efficiently compute $y = D_A(M)$, thus forgery should be computationally infeasible.

Question: Do you buy this argument?