6.857 Computer and Network Security September 27, 2001

Lecture Notes 7 : DSA, Mental Poker, RSA, and Blind Signatures

Lecturer: Ron Rivest *Scribe: Schmidt/Magdsick/Mazza/Baekkelund*

Outline

- Digital Signature Algorithm (DSA)
- Mental Poker – intro
- "RSA" (modulo a prime)
- Mental Poker – protocol
- RSA (modulo a composite)
- Blind Signatures

Overview

We continue forward with our underlying theme of electronic voting. Last lecture, we covered ElGamal, GCD, the discrete logarithm problem, and other number theory. Today, we cover digital signatures (DSA), RSA, and blind signatures while introducing Mental Poker, a neat little problem with interesting analogies that will help us attack our complex problem of secure electronic voting.

1 Digital Signature Algorithm (DSA)

DSA is a public key signature algorithm and is specified by the NIST's Digital Signature Standard[1] (DSS). DSA is used to create a small[2], publicly verifiable signature σ for a given message M.

The DSA has three components, key generation, signature creation, and signature verification.

[1] http://www.nist.gov/public_affairs/releases/digsigst.htm
[2] A small signature is good.

Key Generation:

$q =$ some random 160-bit[3] prime
$p = qt + 1 \quad (1024 \text{ bits})$
g with order q, g not a generator (order q, not $p-1$)
$x \in_R \{0, 1, ..., q-1\}$
$y = g^x (\bmod p)$

public key : (p, q, g, y)
secret key : x

Signature Creation:

for message M calculate $m = h(M)$
$k \in_R \{1, 2, ..., q-1\}$
$r = (g^k \mod p) \mod q$
$s = (m + rx)/k \ (\bmod q)$

$\sigma = (r, s)$, where r and s are each 160 bits

Signature Verification:

given M, p, y, r, s, q, g, h
check if $0 < r < q$ and $0 < s < q$
compute $w = s^{-1} (\bmod q)$

check if $r = g^{wm} y^{rw} \ (\bmod p) \mod q$

Question : *Doesn't DSA specify SHA-1 as the hash function h?*
Answer : Yes, h is SHA-1 in DSA, so $length(m)$ is 160 bits.

Question : *I've heard that DSA has a subliminal channel...*
Answer : Yes, it is true that a malicious manufacturer could use a guess and check algorithm to have some control over the signature output by manipulating k. In this way, a DSA implementation could leak some bits of the secret key in each signature. The malicious manufacturer would only need to know the special encoding format to extract leaked bits of the key.[4]

Question : *What about the tightness or provability of the security of DSA?*
Answer : These schemes used in practice are not provably tight or anything like the theory. In essence, one can only reduce such claims of provability or tightness to "hard" problems like factoring or discrete logarithms.

2 Mental Poker Introduction

Suppose Alice would like to play a fair game of poker with her nemesis, Bob. Since Alice and Bob cannot stand to be near each other, they are forced to play "Mental Poker" over the phone (or

[3] 160-bit numbers are chosen because square root attacks, like the solution to Problem 1-1, would still require brute-forcing through 2^{80} possibilities.

[4] Note that any implementation that allowed a malicious manufacturer to guess k (for instance using a poor pseudorandom number generator) would allow the manufacturer to extract the secret key x from just one signature. Also of note is Karl's clarification of this question and his answer to his own question, emailed out to the class list by the staff.

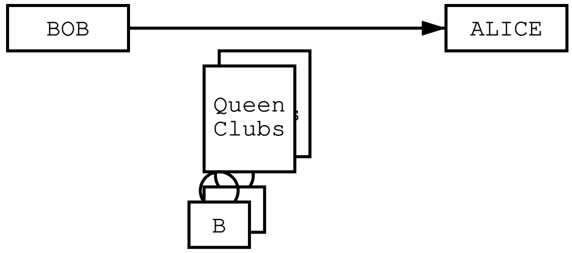

Figure 1: *Bob encrypts, shuffles, and sends all 52 cards to Alice.*

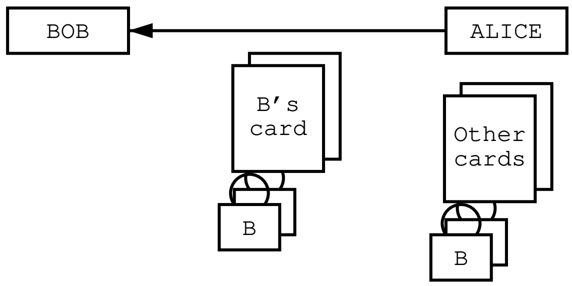

Figure 2: *Alice picks Bob's cards and returns them to him.*

Internet). Mental Poker is an elegant protocol for playing a fair game of poker with an untrustworthy opponent solely by sending one another messages.

The game begins with a "fair deal." First, Bob encrypts all the cards and sends them to Alice in a scrambled order. (See Figure 1.) Next, Alice selects and returns Bob's cards from the shuffled cards sent to her. Bob receives and decrypts his cards. (See Figure 2.) Also, Alice selects and encrypts her cards and sends them to Bob. (See Figure 3.) Bob decrypts Alice's cards and sends them to her. Alice receives and decrypts her cards. (See Figure 4.)

Now, both players have their cards, and neither could know the cards of the other. Play could continue on as in normal poker with players retrieving cards from the rest of the deck with similar protocols to the above methods. At the end of the hand, both players reveal their keys so they can read each other's cards and determine a winner. Note that this depends on the "communtativity"

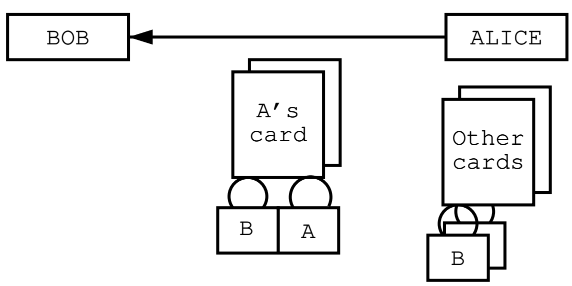

Figure 3: *Alice picks her cards, encrypts them, and sends them to Bob.*

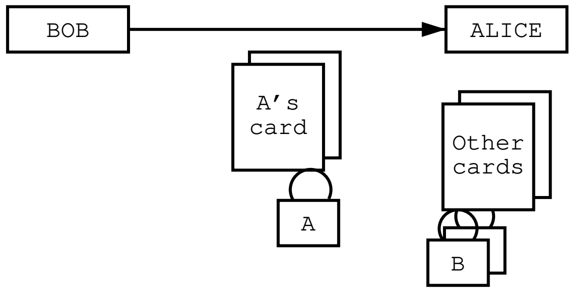

Figure 4: *Bob decrypts Alice's cards and sends them to her.*

of Alice's and Bob's encryption algorithms.

3 "RSA" (modulo a prime)

This Mental Poker cryptosystem idea works perfectly! But can we actually find an encryption function that works this way...? Let's look at a simple version of RSA. (This scheme is actually due to Pohlig.)

Alice and Bob agree on a large, public prime p.

Consider $E(m)$ to be the function $m^e \pmod p$ for some secret key e and message m.

Since $(m^e)^{e'} = (m^{e'})^e$ for secret keys e and e', one can accomplish $D_A(D_B(E_A(E_B(m)))) = m$ like we wanted above.

Question : *When is this one-to-one (i.e. invertible)? ... and when it is, how do we decrypt it?*
Answer : Let's play with logarithms.

"RSA" (modulo a prime): let g be a generator (of order p)
If $m = g^k \pmod p$, then $m^e = g^{ke} \pmod p$

exponentiation by e is equivalent to multiplying the logarithm by e
$m \rightarrow m^e \pmod p$
$k \rightarrow ke \pmod{p-1}$

Question : *Well ... how do we decrypt it?*
Answer : To undo multiplication, multiply by the multiplicative inverse of e (call it d), modulo $(p-1)$. This is only possible if $gcd(e, p-1) = 1$ (verifiable by Extended Euclid Algorithm).

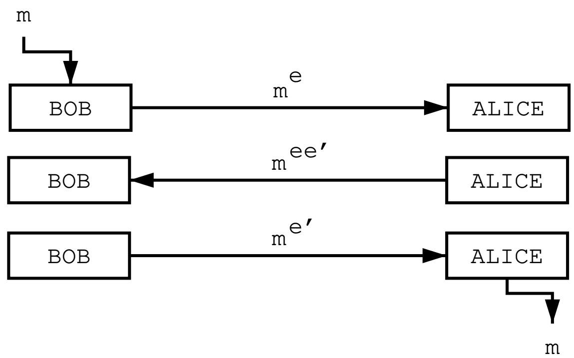

Figure 5: *The Mental Poker protocol.*

"RSA" (modulo a prime): $e \cdot d \equiv 1 \mod (p-1)$

Example:[5]
6^{-1} (mod 16) doesn't exist
3^{-1} (mod 16) $= 11$

Theorem : If p is prime, $gcd(e, p-1) = 1$, $e \cdot d \equiv 1 \pmod{p-1}$
then $(m^e)^d \equiv m \pmod{p}$ holds $\forall m$

In this encryption algorithm, d and e are both private. Hence, it is not a public key cryptosystem. It is an interesting secret key encryption scheme.

Question : *So ... does this solve the poker problem?*
Answer : Yes! Back to poker ...

4 Mental Poker – protocol

Alice and Bob agree on a global, public prime p. Bob has secret encryption and decryption keys e and d, and Alice has secret encryption and decryption keys e' and d' *s.t.*:

$$\begin{aligned} e \cdot d &\equiv 1 \pmod{p-1} \\ e' \cdot d' &\equiv 1 \pmod{p-1} \end{aligned}$$

Using our special "RSA" (modulo a prime) algorithm, we can verify that each of Alice's cards can remain hidden from Bob but be readable by Alice in the end. (See Figure 5.)

[5]The actual example given in class used 15 as the "$p-1$". 16 has many factors. This is a modified example, using $p = 17$, $p-1 = 16$. Note also that both e and d must be odd in RSA or "RSA" (modulo a prime).

Bob encrypts and sends each m_i	$\rightarrow$	$m_i^e = g^{ke} \pmod p$
Alice sends each of Bob's cards back to him		
Bob receives and decrypts his cards		
Alice encrypts and sends her cards	$\rightarrow$	$m_A^{ee'} = g^{kee'} \pmod p$
Bob decrypts and returns her cards	$\rightarrow$	$m_A^{ee'd} = g^{kee'd} \pmod p$
Alice receives and decrypts her cards	$\rightarrow$	$m_A^{ee'dd'} = g^{kee'dd'} \pmod p$

Since $m^{ee'dd'} \pmod p = m^{(e'd')(ed)} \pmod p$ and $x^{ed} \pmod p = x \pmod p$, then $m^{(e'd')(ed)} \pmod p = m^{e'd'} \pmod p$. Similarly, $m^{e'd'} \pmod p = m \pmod p$, and Alice has her cards.

5 RSA (modulo a composite)

RSA was the first public key digital signature proposed. The space of elements for the message we want to encrypt and for the ciphertext are both the same: $\{0, 1, ..., n-1\}$ where $n = p \cdot q$ is the product of two randomly chosen large prime numbers p and q.

The Chinese Remainder Theorem tells us that everything happens in the field of $\pmod n$ happens simultaneously in the field $\pmod p$ and in the field $\pmod q$. In particular, if the following two properties are verified:

$m^{ed} \equiv m \pmod m$
$m^{ed} \equiv m \pmod q$

Then

$m^{ed} \equiv m \pmod n$

(More generally, if $x \equiv y \pmod p$ and $x \equiv y \pmod q$ then $x \equiv y$ $(mod\ (p \cdot q))$.)

The three steps of RSA are:

Key Generation:	Generate two random primes numbers p, q $s.t.$ $p \neq q$ Compute $n = p \cdot q$ Pick random e $s.t.$ $1 < e < (p-1) \cdot (q-1)$ and $gcd(e, (p-1), (q-1)) = 1$ Compute d $s.t.$ $1 < d < (p-1) \cdot (q-1)$ and $e \cdot d \equiv 1 \pmod{(p-1) \cdot (q-1)}$ The public key is (n, e) and the private key is d.
Signature Generation:	Compute $\sigma = m^d \pmod n$ The signature for m is σ
Signature Verification:	Given the public key (n, e) Verify that $\sigma^e \pmod n = m$

The security of this algorithm is completely dependent upon the difficulty of the mathematical problem of factorization[6]. If an adversary were able to factor n to $(p-1) \cdot (q-1)$, then using the Extended Euclidean Algorithm, the adversary could easily calculate d from $(p-1) \cdot (q-1)$ and e. Even today, twenty years after RSA's conception, the best factoring algorithm only has the following running time $e^{(1.93 \cdot (ln\, n)^{1/3} \cdot (ln\,(ln\, n))^{2/3})}$ and thus cracking RSA remains computationally infeasible for large values of n.

One last note is that the multiplicative property of this signature scheme is a double-edged sword. While it can help us solve problems like that of Mental Poker, at times the multiplicative property can cause trouble when an exact multiple of a message would still make sense in the message domain.

6 Blind Signatures

The goal of a blind signature is to have an authority sign a message without the authority actually seeing the message it has signed. One immediate advantage of this system is that the signer is later unable to associate a given signed message with the sender.[7]

The more apparent applications of this type of scheme involve preserving some sort of anonymity on the part of the sender while still providing assurances that the message will be verifiable. Two concrete examples of involving blind signatures are voting (where the goal is preserve the voter to vote privacy) and electronic cash (where one cannot track the path of cash beyond where one got it).

A blind signature is somewhat of a misnomer. A better analogy is that of a stamp of approval, as the authority is only really giving approving the message, not agreeing to some terms. Understandably, limits on the possible types of messages validated by an authority's blind signature must be set, as one could send to the authority a contract where the authority sells their soul. The authority would blindly sign it, not knowing what the message was. This could potentially mess up some things pretty badly.

Suppose that Alice wants to have a message M blindly signed by Bob. We choose an appropriate hash function and let $m = h(M)$. The blind signature scheme is as follows:

Bob has the following secret parameters (d, p, q) and posts the public values $(n = p \cdot q, e)$ $s.t.$ $e \cdot d \equiv 1 \pmod{n}$. Alice chooses a random number r $s.t.$ $0 <= r <= n-1$ and $gcd(n, r) = 1$ and sends to Bob $m \cdot r^e \pmod{n}$. Bob sends back to Alice $m^d \cdot r^{de} = m^d \cdot r \pmod{n}$. As she knows r, Alice can now decompose Bob's message (by multiplying by $r^{-1} \bmod n$) and get m^d, Bob's blind signature on the original message.

[6] However, no one has shown that breaking RSA implies factoring is easy.
[7] This assumes that the authority sees more than one message!