

CS556 Introduction to Cryptography - Prof. Ming-Deh Huang
Scribe: Iftikhar A Burhanuddin
burhanud@usc.edu
Class #11, #12 - October 1, 3, 2002

Announcements:

Notes for classes #10 and #11 are online. HW #2 is due on October 8.

Topics for today:

1. Public Key Certification
2. Attacking RSA
3. Chosen Ciphertext Attack
4. Protocol Failure
5. Blinded Signatures

1. Public Key Certification

In an ideal scenario all public keys would be placed in a directory accessible by all. If Alice wanted to send an encrypted message to Bob, she would look up Bob's public key E_B in the directory and apply it to her plaintext and send it over to him. But in a less than ideal world such as ours nothing would stop Eve from going and changing Bob's public key to something else or maybe even replacing E_B with E_E her own key so that she could read all messages addressed to Bob. How can a correspondence be established between Bob and his key $E_B : B \leftrightarrow E_B$? The concept of the *Certification Authority [CA]* provides a viable solution.

Bob would go to a CA like Verisign and apply for a public key. Versign would then issue Bob a certificate $(T \parallel ID_B \parallel E_B)$, where T is a time stamp, ID_B has information about Bob, like name, address, etc., and E_B is the public key for Bob. To convince a third person like Charlie that Bob's key has been issued by a trusted party, Versign would sign the certificate with its private key D_A and give Bob $D_A(T \parallel ID_B \parallel E_B)$. Charlie would then apply Versign's public key E_A to unscramble the signed certificate and retrieve Bob's key E_B as $E_A(D_A(T \parallel ID_B \parallel E_B)) = (T \parallel ID_B \parallel E_B)$.

So the CA paradigm associates Bob with his key E_B and as vice-versa as follows:

1. A central figure whose public key is recognized by everybody takes over as the CA.
2. The CA assigns keys to Alice, Bob, etc.

Signing is a very important feature of Public Key Cryptography. We just saw the use of the secret/decryption/private key to sign certificates, next we'll look at an instance of signing messages.

To send message m to Bob, Joe sends the following over $E_B(m \parallel D_J(P_J))$, where P_J says something like "This is Joe. October 1, 02." When Bob receives the ciphertext he applies his private key D_B to obtain $(m \parallel D_J(P_J))$. He then reads the message and verifies that the message originated from Joe by recovering P_J using E_J .

Certifying keys $B \leftrightarrow E_B$ implies not just sending messages to Bob securely but also allows Bob to sign her messages and others to verify authenticity of messages coming from Bob.

Can Joe find out if the message m has been tampered with or altered? Also what prevents Eve from using Joe's signature $(m \parallel D_J(P_J))$ and fooling Bob about the origin of her message? How can this be remedied?

2. Attacking RSA

Clearly **Factoring** $n \Rightarrow$ **computing** $\phi(n)$. Also last time we saw how the following theorem

Thm. *Given an integer m which satisfies $a^m \equiv 1 \pmod{n}$ for all $a \in \mathbf{Z}/n\mathbf{Z}^*$ we can factor n in random polynomial time* with its corollary

Cor. When $m = \phi(n)$

gives us a constructive method to compute the factors of n knowing $\phi(n)$ in random polynomial time. Hence **computing** $\phi(n) \Rightarrow$ **Factoring** n . Another way of saying this is that computing $\phi(n)$ is random polynomial time **equivalent** to factoring n .

But there are other things apart from the above which could make a RSA based cryptosystem vulnerable such as protocol failure, chosen ciphertext attack, semantic security.

3. Chosen Ciphertext Attack

Inversion Problem. Given y and E to compute x such that $y = E(x)$, i.e., to compute $x = D(y)$ but D is secret.

If E and D are the corresponding RSA functions we have the **RSA problem**.

Chosen Ciphertext Attack. Suppose in addition to y and E we can choose $y' \neq y$ and ask for $x' = D(y')$.

In this model an oracle Ω is available to the adversary which honors one request for decryption - given y the oracle returns $D(y)$.

If the system is secure under this attack it is called *chosen ciphertext secure*. The security guaranteed under this model is stronger than the one guaranteed under the inversion problem.

Is RSA secure under chosen ciphertext attack? No.

Reasoning. Let's play the role of the adversary. Say we pick a random z such that $y' \neq y$ where $y' := yE(z)$. So y' is our chosen ciphertext and the oracle Ω decrypts it for us and let $x' := \Omega(y') = D(y')$. $D(y') = D(y)D(E(z)) \Rightarrow x' = D(y)z \Rightarrow D(y) = z^{-1}x'$. Hence as we know the RHS we can compute the decryption of y .

Though in a more realistic model we'll have lots of plaintext-ciphertext pairs the above chosen ciphertext attack security model is not a purely academic exercise. Can you think of a scenario where such a security guarantee will be necessary?

4. Protocol Failure

There is a danger of a signed document being misused, illustrated by the following scenario. Alice gives a signed message ("Dear Bank. Please give Bob \$100") to Bob. Bob reuses this signed message over and over again till his heart's content. Possible solutions to stop his misuse are introducing a check # and/or time-stamping so that the validity of the message expires in appropriate time. We'll next present two more cases of Protocol failure where the issues are not due to the inherent strength/weakness of RSA but implementation lapses.

4.1 Small Domain

Say we encrypt our message letter-by-letter using a RSA encryption function. If an adversary knew about our encryption technique all he would have to do to decipher messages is to construct a table of the letters of the al-

phabet and their corresponding encryptions and do a reverse lookup. More generally if our messages came from a very small subset of our plaintext set $\mathbf{Z}/n\mathbf{Z}$ (which perse is quite big) an adversary would be able to easily break the system.

Also if the documents we want to encrypt are short and we left pad it with zeros and say our plaintexts end up being equivalent to numbers less than 10^{10} with n being a 256 digit integer, the adversary will easily be able to enumerate all the 10^{10} possibilites and read all the ciphertext.

Hence there is a need for our encoding techniques to be sufficiently randomized so that we obviate the case of plaintext messages coming from a tiny domain.

4.2 Same modulus

If two users A and B use the same modulus the system is vulnerable as follows: Say $E_A = (e_1, n)$ and $E_B = (e_2, n)$. Suppose C sends the same plaintext x 'securely' is sent to both A and B , that is, A receives $y_1 = x^{e_1} \bmod n$ and B receives $y_2 = x^{e_2} \bmod n$. An adversary E who is snooping on the network can decrypt x as follows:

e_1, e_2 are random numbers between 1 and $n - 1$. Suppose $\gcd(e_1, e_2) = 1$ (the gcd being not equal to 1 is very unlikely). E can compute c, d using the euclidean gcd algorithm such that $ce_1 = 1 + de_2$. And $y_1^c = x^{e_1c} = x.x^{de_2} = xy_2^d \Rightarrow x \equiv y_1^c(y_2^d)^{-1} \bmod n$.

Aside. On the other hand say $\gcd(e_1, e_2) \neq 1$ (very small chance) but instead $\gcd = 2$ then we end up with $x^2 \equiv y_1^c(y_2^d)^{-1} \bmod n$ which is not an easy problem to solve as you've seen in HW #2.

Therefore care should be taken that no two users end up with the same modulus. This gives us another reason why n needs to be sufficiently big so that there is an abundant supply of primes.

With the same token a *lazy* user who reuses his modulus puts the security of his messages at jeopardy.

5. Blinded Signatures

Blinded signatures are interesting variations of signing documents, where you want somebody to sign a message without him/her knowing what he/she is signing.

Say Bob works for counter intelligence agency aka c.i.a and he wants to

pick up a name for his undercover ops and decides on “Raging Bull”. Bob wants the c.i.a to sign a message m which says “whoever uses the cover of *Raging Bull* should be given diplomatic immunity. Signed Director c.i.a”. Bob wants the c.i.a to blindly sign his message without the clerks/computers knowing his covername. Let A stand for the central intelligence Agency

We’ll again use the trick of randomizing a message leveraging the multiplicative nature of E_A and D_A help Bob get a blind signature on m from the Agency.

1. Bob computes $m' = mE_A(r)$ where m is the message and r is a random number picked by Bob. $E_A(r)$ is called the blinding factor.
2. Bob gives m' to the Agency for signing and gets back $D_A(m')$
3. Bob extracts $D_A(m)$ as follows:

$$D_A(m') = D_A(mE_A(r)) = D_A(m)D_A(E_A(r)) = D_A(m)r \Rightarrow D_A(m) = r^{-1}D_A(m')$$

Instead of the getting message m signed Bob wants to get the following message signed by the Agency: “I the c.i.a director retire Bob from service and henceforth he’ll be paid 10k per diem.” How does the c.i.a prevent Bob from cheating while signing blinded documents? Read 5.3 of Schenier for possible fixes.

Remark: The multiplicativity of $D(x)$ forms the basis for blinded signatures and also the 1% – 100% attack. These are two sides of the same coin.

We can dream up applications like Blinded Voting. Please browse through the Applied Cryptography book as we are in a position to read about crypto applications like Digital cache, etc.

Remark.

1. Does RSA succumb to quantum computers because it is weak or due to the power of quantum computing?
2. We know how to add, multiply (though we don’t know the most efficient way to multiply), compute gcd, etc. It seems like a humble request to be able to factor integers which is another elementary operation. But there are no *known* polynomial time algorithms for factoring. On the other hand the existence of sub exponential time algorithms for Integer

factoring seems to suggest that factoring is not difficult as Travelling Salesman Problem (TSP). Also if we prove that there is no polynomial time factoring algorithm we've solved the $P \neq NP$ question in the affirmative.