CPSC 418/MATH 318 Introduction to Cryptography

More on RSA, Probabilistic encryption, El Gamal PKC, Provable Security, More Number Theory

Renate Scheidler

Department of Mathematics & Statistics
Department of Computer Science
University of Calgary

Week 9

(March 14 is Pi Day)

Outline

Modular Inverses

Recall that $\mathbb{Z}_m^* = \{a \in \mathbb{Z}_m \mid \gcd(a, m) = 1\}$ is the set of integers between 1 and m that are coprime to m.

$\mathbb{Z}_m^*$ consists of exactly those integers that have *modular inverses*:

- for every $a \in \mathbb{Z}_m^*$, there exists $x \in \mathbb{Z}_m^*$ such that $ax \equiv 1 \pmod{m}$.

In the RSA set-up to generate user keys, given $e \in \mathbb{Z}_{\phi(n)}^*$, we need to find $d \in \mathbb{Z}_{\phi(n)}^*$ such that

$$ed \equiv 1 \pmod{\phi(n)} .$$

Computing Modular Inverses

Given $a \in \mathbb{Z}_m^*$, solve the linear congruence $ax \equiv 1 \pmod{m}$ for $x \in \mathbb{Z}_m^*$.

- We want x such that

$$m \mid ax - 1 \iff ax - 1 = ym \iff ax - my = 1$$

  for some $y \in \mathbb{Z}$.
- Can be solved using the Extended Euclidean Algorithm.
- We only need to compute the B_i because we only need x, not y.
- When using Bezout's method, be sure to start with $a = q_0 m + r_0$, even if $a < m$ (so $q_0 = 0$), else you get the wrong count for the number n of division steps.
- Always check your answer!

A Back Substitution Example

Solve $95x \equiv 1 \pmod{317}$ for $x \pmod{317}$ using back substitution.

Here $a = 95$ and $m = 317$. The Euclidean algorithm yields

$$\begin{aligned} 95 &= 317 \cdot 0 + 95 \\ 317 &= 95 \cdot 3 + 32 \quad (1) \\ 95 &= 32 \cdot 2 + 31 \quad (2) \\ 32 &= 31 \cdot 1 + 1 \quad (3) \\ 31 &= 1 \cdot 31 + 0 \end{aligned}$$

Thus, $\gcd(95, 317) = 1$ and back substitution yields

$$\begin{aligned} 1 &\overset{(3)}{=} 32 - 31 \overset{(2)}{=} 32 - (95 - 2 \cdot 32) = 3 \cdot 32 - 95 \\ &\overset{(1)}{=} 3(317 - 3 \cdot 95) - 95 = 3 \cdot 317 + (-10) \cdot 95 \ . \end{aligned}$$

So $1 \equiv (-10) \cdot 95 \pmod{317}$ and hence $x \equiv -10 \equiv 307 \pmod{317}$.

Same Example Using Bezout's Method

Solve $95x \equiv 1 \pmod{317}$ for $x \pmod{317}$ using Bezout's method.

Recall the Euclidean algorithm for $a = 95$ and $b = 317$:

$$\begin{aligned} 95 &= 317 \cdot 0 + 95 & q_0 &= 0 \\ 317 &= 95 \cdot 3 + 32 & q_1 &= 3 \\ 95 &= 32 \cdot 2 + 31 & q_2 &= 2 \\ 32 &= 31 \cdot 1 + 1 & q_3 &= 1 \\ 31 &= 1 \cdot 31 + 0 & q_4 &= 31 \end{aligned}$$

So $n = 4$ and our solution will be $x \equiv (-1)^{4-1} B_{4-1} \equiv -B_3 \pmod{317}$.

The Bezout tableau for the quantities B_k is

		0	3	2	1
1	0	1	3	7	10

So $x \equiv -10 \equiv 307 \pmod{317}$.

RSA – Recap

Public key: (n, e); private key d where

- $n = pq$ for distinct primes p, q;
- $1 < e < n$, $\gcd(e, \phi(n)) = 1$;
- $ed \equiv 1 \pmod{\phi(n)}$

and $\phi(n) = (p-1)(q-1)$.

Encrytion of $M \in \mathbb{Z}_n^*$: $C \equiv M^e \pmod{n}$.

Decryption of $C \in \mathbb{Z}_n^*$: $M \equiv C^d \pmod{n}$.

Efficiency of RSA

Set-up (need only be done once):

- Prime generation uses a pseudo-random number generator (PRNG), followed by a probable primality test (like the Fermat test).
- Generating e again requires a PRNG and one gcd calculation (EA) – or just pick your favourite e.
- Computing n and $\phi(n)$ is negligible.
- Computing d requires finding a modular inverse (EEA)

Encryption and decryption: modular exponentiation (like Diffie-Hellman).

Security of RSA

RSA Problem (extracting e-th roots modulo n):

Given e, n and $C \in \mathbb{Z}_n^*$, find $M \in \mathbb{Z}_n^*$ with $M^e \equiv C \pmod{n}$.

Integer Factorization Problem (IFP):

Given an integer $N > 1$, find a non-trivial factor of N.

- If an adversary can solve an instance of the IFP, she can solve the RSA problem (by factoring n and finding the private key d in the same way as the designer).
- It is unknown if there are ways of solving the RSA problem without factoring (or solving one of the other equivalent problem listed below).

Total Breaks of RSA

The following approaches break RSA (assume (e, n) is known):

Factoring n, i.e. finding p, q
$\downarrow \phi(n) = (p-1)(q-1)$ $\quad\quad$ $\uparrow$ Solve $x^2 - (n - \phi(n) + 1)x + n = 0$ for x
Finding $\phi(n)$
$\downarrow$ Solve $ed \equiv 1 \pmod{\phi(n)}$ $\quad$ $\uparrow$ See Algorithm 6.10 in Stinson-Paterson
Finding the private key d

Note:

- The quadratic equation above has two solutions, namely p and q.
- There is an efficient algorithm that given any multiple of $\phi(n)$ finds $\phi(n)$ with high probability. Note that $ed - 1$ is such a multiple.

Total Breaks of RSA (cont'd)

All three approaches on the previous slide are computationally equivalent:

- if one can be achieved, any of the other two one can be achieved with very little computational overhead.
- so there are *three* equally good trapdoors here: $\{p, q\}$, $\phi(n)$ and d.

There is no proof that RSA is secure!

- No proof that factoring is hard
- Not proved that other methods to solve the RSA problem exist which do not rely on factoring (i.e. not known whether breaking RSA is *equivalent* to factoring n)

In any case, we need to design RSA systems such that $n = pq$ cannot be factored easily.

Factoring Record

The fastest known factoring algorithm is again the Number Field Sieve (slightly different from the DLP NFS, but invented first). Run time:

$$\exp\left(c(\log n)^{1/3}(\log\log n)^{2/3}\right) = n^{c(\log\log n/\log n)^{2/3}}$$

with

$$c = \sqrt[3]{64/9} = 1.92\ldots$$

Current RSA modulus factoring record: RSA-250 (250 decimal digits, 831 bits): Boudot-Gaudry-Guillevic-Heninger-Thomé-Zimmerman (February 2020, same people as the DLP record)

$$\begin{aligned} & 2140324650240744961264423072839333563008614715144755017797754920881418023447140136643345519095804679610992851872470914587687396262192155736304745477052080511905649310668769159001975940569345745223058932597669747168173806936489469987157849497593749793764132589477071580278790190170577389084825014742943447208116859632024532344630238623598752668347708737661925585694639798853367 \\ = & 337320275949781565562260106053551142279407603447675546667845209870238417292100370802574486732968818775657189862580369320627 11 \end{aligned}$$

2700 core years with Intel Xeon Gold 6130 CPUs 2.1GHz as reference

See `https://en.wikipedia.org/wiki/RSA_Factoring_Challenge`

Choice of RSA Parameters

Requirements for p and q:

1. Probable primes with high probability (say 2^{-100}) — use a good probabilistic primality test.
2. Large: at least $2^{1536} \approx 10^{463}$ (so n is 3072 bits)
3. Not too close together; $|p-q| > 2^{128}$ for $p, q \approx 2^{1536}$
4. p and q must be *strong* primes, i.e. $p-1$, $q-1$, $p+1$, $q+1$ all have a large prime factor (see p. 291 of the *Handbook of Applied Cryptography*).

   E.g. pick a Sophie Germain prime p' (so $p = 2p'+1$ is a safe prime) so that $(p+1)/4 = (p'+1)/2$ is prime or has a large prime factor; same for q.
   - May also want to choose $p'-1$ to have a large prime factor to avoid *cycling attacks* (where a modest number of repeated encryptions "cycle back" to the plaintext)

   Choosing random p, q may be sufficient (Rivest-Silverman 1999)

Choice of RSA Parameters (cont'd)

Requirement for e:

- For efficiency reasons, e is often chosen small; a popular choice is $e = 2^{16}+1 = 65537$ (great for binary exponentiation, only two '1' bits).
- Beware of really small e for certain applications!
- In practice, can use $e = 3$, but *only* when RSA is used in conjunction with a secure padding mechanism (eg. OAEP — next week!)

Requirement for d:

- $d \lessapprox n^{0.25}$ (Wiener, 1990)
- $d \lessapprox n^{0.292}$ (Boneh & Durfee 2000, extension of Wiener's attack)

Attack Scenario: Common Modulus, Common Message

Two users with public keys (e_1, n) and (e_2, n) where $\gcd(e_1, e_2) = 1$.

Alice sends both users the same message M, encrypted:

$$C_i \equiv M^{e_i} \pmod{n}, \qquad (i = 1, 2).$$

The following attack finds M:

1. Use the Extended Euclidean Algorithm to obtain integers x, y with

   $$xe_1 + ye_2 = 1.$$

2. Then

   $$C_1^x C_2^y \equiv (M^{e_1})^x (M^{e_2})^y \equiv M^{e_1 x + e_2 y} \equiv M^1 \equiv M \pmod{n}.$$

Attack can be extended to more than two users sharing the same modulus.

Attack Scenario: $e = 3$, Common Message

Three users with public keys $(3, n_1)$, $(3, n_2)$, $(3, n_3)$, $\gcd(n_1, n_2, n_3) = 1$. (Note that if $\gcd(n_i, n_j) > 1$, then n_i and n_j are factored!)

Alice sends all three users the same message M, encrypted:

$$C_i \equiv M^3 \pmod{n_i}, \qquad (i = 1, 2, 3).$$

The following attack finds M:

1. Use the Extended Euclidean Algorithm to obtain integers u, v, w with

   $$un_2n_3 + vn_1n_3 + wn_2n_2 = 1.$$

   (Apply EEA first to n_1 and n_2, then to n_1n_2 and n_3.
2. Put

   $$C \equiv un_2n_3C_1 + vn_1n_3C_2 + wn_1n_2C_3 \pmod{n_1n_2n_2}.$$

   Then

   $$C \equiv C_i \equiv M^3 \pmod{n_i} \qquad (i = 1, 2, 3).$$

Attack Scenario: $e = 3$, Common Message (Cont'd)

3. Now we have

$$0 < C < n_1 n_2 n_3$$
$$0 < M^3 < n_1 n_2 n_3$$
$$C \equiv M^3 \pmod{n_1 n_2 n_3}$$

This implies that $C = M^3$ and hence $M = \sqrt[3]{C}$.

Attack can be extended to any e and at least e users.

Attack Scenario: $e = 3$, Three Similar Messages

Suppose a user has RSA public key $(3, n)$.

Alice sends three messages that differ in a few places in a known way to that user. Then these messages can be efficiently found by an adversary using linear algebra modulo n.

Attack can be extended to any e and at least e similar messages with known differences.

Wiener's Attack on Small d

Assume $d < \sqrt[4]{\frac{n}{36}} \approx 0.408\sqrt[4]{n}$ and $q < p < 2q$.

$ed = 1 + k\phi(n)$ where $d, k, \phi(n)$ are unknown. Can show that $1 \le k < d$.

$$0 < kn - ed = k(n - \phi(n)) - 1 < d(n - \phi(n)) - 1 < 3d\sqrt{n}.$$

Divide by nd:

$$0 < \frac{k}{d} - \frac{e}{n} < \frac{3}{\sqrt{n}} < \frac{3}{6d^2} = \frac{1}{2d^2} .$$

By the theory of *convergents*, it turns out that under these conditions

$$k = A_i , \qquad d = B_i ,$$

where A_i, B_i are the Bezout sequences arising from applying the Euclidean algorithm applied to e and n (easy to compute!)

Multiplicative Attacks on RSA

"Textbook" RSA is not secure against *multiplicative* attacks.

Multiplicative (or *homomorphic*) property of RSA:

$$(M_1 M_2)^e \equiv M_1^e M_2^e \equiv C_1 C_2 \pmod{n}$$

i.e. the encryption of a product is the same as the product of the encryptions.

This means that a factorization of the plaintext implies one of the corresponding ciphertext, which can be exploited in two attacks.

Adaptive CCA on RSA

An attacker wishing the decryption M of some RSA ciphertext C proceeds as follows:

1. Generates $X \in \mathbb{Z}_n^*$ with $X^e \not\equiv 1 \pmod{n}$.
2. Computes $C' \equiv CX^e \pmod{n}$ (this is the chosen ciphertext; note that $C' \neq C$).
3. Obtains the corresponding plaintext

$$M' \equiv (C')^d \equiv C^d (X^e)^d \equiv MX \pmod{n}$$

4. Computes $M \equiv M'X^{-1} \pmod{n}$, where X^{-1} is the inverse of $X \pmod{n}$

Meet-in-the-Middle Attack on RSA (Passive)

If $M \approx 2^k$ for some bit length k, then with non-negligible probability, M is composite and satisfies $M = M_1 M_2$ with $M_1, M_2 \approx 2^{k/2}$.

- The probability that a number of $40 - 64$ bits factors into equal-size factors is between 18 and 50 percent (see Table 1 of "Why textbook El Gamal and RSA encryption are insecure (extended abstract)" by Boneh, Joux, and Nguyen, in ASIACRYPT 2000)).

The adversary builds a list $\mathcal{L} = \{1^e, 2^e \pmod{n}, \ldots, \lfloor 2^{k/2} \rfloor^e \pmod{n}\}$ and a list $\mathcal{L}'$ of all their inverses $\pmod{n}$.

- She then computes $C(i^*)^e \pmod{n}$ $(i^* = 1, 2, 3, \ldots)$ and searches for a match $j^e \in \mathcal{L}$ (here $(i^*)^e$ is the inverse of i^e modulo n).
- If $C(i^*)^e \equiv j^e \pmod{n}$ for some j , then $C \equiv (ij)^e \pmod{n}$ and hence $M \equiv ij \pmod{n}$.

Requires $2 \cdot 2^{k/2}$ modular exponentiations (rest is negligible).

Example Application of Meet-in-the-Middle

Hybrid encryption: consider the case where 2048-bit RSA modulus is used to encrypt a 56-bit DES key.

- Here, $k = 56$ and each $i^e \pmod{n}$ takes about $\log_2(n) \approx 2048$ bits of storage
- The list requires $2^{28} \cdot 2048 = 2^{39}$ bits of storage (about 64 GB)
- Requires 2^{29} modular exponentiations.
- This is easily done on a PC.

Protecting Against the Multiplicative Property

The multiplicative property of RSA can be obscured by imposing a format on plaintexts and then randomizing the formatted text.

Can defeat CCA by rejecting decryptions of "invalid" messages.

One example is RSA-OAEP (discussed later):

- RSA plus optimal asymmetric encryption padding
- plaintext is padded with 0's and transformed to a statistically random bit string via a reversible, randomized, unkeyed transformation.

Advantages of RSA

Advantages:

1. Seems to be mathematically secure.
2. Key size is "relatively" small — two 463-digit numbers — although other PKCs have smaller keys (eg. elliptic curve systems).
3. No message expansion — ciphertexts and plaintexts have the same length.
4. Can be used as a signature scheme (covered later).

Disadvantages of RSA

Disadvantages:

1. Very slow compared to 3DES, AES, and other symmetric key cryptosystems. Decryption is also slower than elliptic curve based systems.
2. Finding keys is fairly expensive.
3. Security is unproved
   - See also the Etherium Foundation's website on further RSA security assumptions and cash bounties at `https://rsa.cash`
4. "Textbook" version leaks information and is vulnerable to a number of attacks.

Probabilistic Encryption

One disadvantage of deterministic PKCs is that identical messages always encrypt to the same ciphertext (like block ciphers in ECB mode).

- particularly problematic if the message space is small (*e.g.* electronic yes/no vote)

Probabilistic or *randomized encryption* utilizes randomness to attain a provable, stronger level of security.

As a result, every message can have many possible encryptions, so a small message space is no longer a problem.

- leads to the notion of *semantic* security.

The El Gamal PKC

T. El Gamal (1985): randomized, security based on DLP — alternative to RSA which was based on the integer factorization problem (IFP)

Set-up: the designer produces her public and private keys as follows:

1. Selects a large prime p and a primitive root g of p
2. Generates a random integer x with $1 < x < p-1$ and computes $y \equiv g^x \pmod{p}$ where $1 \leq y \leq p-1$.

Public key: (p, g, y)
Private key: $\{x\}$

Note: multiple users may use the same g and p, but everyone should have their own pair (x, y).

El Gamal Encryption

Messages for the designer are integers M, $0 < M < p$ (so $M \in \mathbb{Z}_p^*$).

To send M encrypted, proceed as follows:

1. Select a random $k \in \mathbb{Z}$, $0 < k < p-1$.
2. Compute and send (C_1, C_2) where

$$C_1 \equiv g^k \pmod{p}, \quad 0 < C_1 < p,$$
$$C_2 \equiv My^k \pmod{p}, \quad 0 < C_2 < p \ .$$

El Gamal Decryption

To decrypt (C_1, C_2), the designer computes

$$\begin{aligned} C_2 C_1^{p-1-x} &\equiv (My^k)(C_1^{p-1-x}) \\ &\equiv (Mg^{xk})(g^{k(p-1-x)}) \\ &\equiv Mg^{xk+k(p-1)-kx} \\ &\equiv M(g^{p-1})^k \\ &\equiv M \pmod{p} \ . \end{aligned}$$

Think of C_1 as a "clue" that can be used to remove the "mask" y^k in C_2, thus "unmasking" the encrypted message M.

An El Gamal Toy Example

$p = 53$, $g = 2$; $x = 14$, $y \equiv 2^{14} \equiv 7 \pmod{53}$.

Private key: $\{14\}$; Public key $(53, 2, 7)$

Encryption of $M = 10$ under public key $(53, 2, 7)$ is (C_1, C_2) where

- the random number selected is $k = 6$
- $C_1 \equiv 2^6 \equiv 11 \pmod{53}$
- $C_2 \equiv 10 \cdot 7^6 \equiv 49 \pmod{53}$

Decryption of $(C_1, C_2) = (11, 49)$ under private key 14 is

$$49 \cdot 11^{53-1-14} \equiv 49 \cdot 11^{38} \equiv 10 \pmod{53} \ .$$

Summary of El Gamal

As with DH key establishment, the security of this system relies on the presumed difficulty of the DLP, but it is unknown whether there are other ways of breaking El Gamal.

Disadvantages:

- Message expansion by a factor of 2 (ciphertext is twice as long as the plaintext).
- Twice as much computational work for encrypting as RSA:
  - two exponentiations (and one multiplication), as opposed to one exponentiation only for RSA.
- A new random number k must be generated for each message.

Advantages: randomized, different security assumption, works in other settings (eg. elliptic curves)

Polynomial Security

Definition 1 (Polynomial security, IND-CPA security)

A PKC is said to be *polynomially secure* or *IND-CPA secure* if no passive adversary can in expected polynomial time select two plaintexts M_1 and M_2 and then correctly distinguish between encryptions of M_1 and M_2 with probability significantly greater than $1/2$.

IND-CPA: indistinguishability under chosen plaintext attacks.

Semantic Security

Definition 2 (Semantic security)

A PKC is said to be *semantically secure* if for all probability distributions over the message space, anything that can be computed by a passive adversary in expected polynomial time about the plaintext given the ciphertext can also be computed in expected polynomial time without the ciphertext.

Intuitively, semantic security is a weaker version of perfect secrecy

- an adversary with polynomially-bounded computational resources (as opposed to infinite resources in perfect security) can learn nothing about the plaintext from the ciphertext.

Equivalence

Theorem 1

A PKC is semantically secure if and only if it is polynomially secure.

Although El Gamal is randomized, it is *not* semantically secure as presented here (see Assignment 3).

We will soon look at a PKC that is semantically secure assuming that a certain number theoretic problem (not DLP or IFP) is hard. But first, we need a bit more number theory.