Public Key Crypto: RSA

Public Key Cryptography: ElGamal and RSA

From *The Economist* Sept 15, 2018, page 34

Article Title: Whack a Mole: The new president (of Columbia) calls off talks with a lesser-known leftist insurgent group.

From *The Economist* Sept 15, 2018, page 34

Article Title: Whack a Mole: The new president (of Columbia) calls off talks with a lesser-known leftist insurgent group.

Context: In 2016 FARC, a left-wing insurgent group in Columbia, signed a peace treaty that ended 50 years of conflict Yeah! The former president of Columbia got the Nobel Peace Prize (the leader of FARC did not – I do not know why). However a more extreme insurgent group, ELN, is still active. Why did FARC negotiate but ELN did not?:

From *The Economist* Sept 15, 2018, page 34

Article Title: Whack a Mole: The new president (of Columbia) calls off talks with a lesser-known leftist insurgent group.

Context: In 2016 FARC, a left-wing insurgent group in Columbia, signed a peace treaty that ended 50 years of conflict Yeah! The former president of Columbia got the Nobel Peace Prize (the leader of FARC did not – I do not know why). However a more extreme insurgent group, ELN, is still active. Why did FARC negotiate but ELN did not?:

Quote: . . . And the ELN's strong *encryption system* has prevented the army from extracting information from seized computers, as it did with FARC.

From *The Economist* Sept 15, 2018, page 34

Article Title: Whack a Mole: The new president (of Columbia) calls off talks with a lesser-known leftist insurgent group.

Context: In 2016 FARC, a left-wing insurgent group in Columbia, signed a peace treaty that ended 50 years of conflict Yeah! The former president of Columbia got the Nobel Peace Prize (the leader of FARC did not – I do not know why). However a more extreme insurgent group, ELN, is still active. Why did FARC negotiate but ELN did not?:

Quote: . . . And the ELN's strong *encryption system* has prevented the army from extracting information from seized computers, as it did with FARC.

Caveat: The article did not say what system they used. Oh Well

Public Key Cryptography: ElGamal

Recall Diffie-Helman

1. Alice and Bob end up sharing a secret.
2. p, g are public keys.
3. Under a hardness assumption Eve does not know the secret.
4. The secret is *not* in Alice or Bob's control

DH cannot be used for the following:

Alice takes the message Lets do our Math/CMSC 456 HW on time this week for a change encrypt it, send it to Bob, and Bob Decrypts it.

We describe the ElGamal Public Key Encryption Scheme where Alice and Bob can encrypt and decrypt under a hardness assumption.

ElGamal is DH with a Twist

1. Alice and Bob do Diffie Helman.
2. Alice and Bob share secret $s = g^{ab}$.
3. Alice and Bob compute $s^{-1} \pmod{p}$.
4. To send m, Alice sends $c = ms \pmod{p}$
5. To decrypt, Bob computes $cs^{-1} \equiv mss^{-1} \equiv m \pmod{p}$

We omit discussion of Hardness assumption (HW)

ElGasarch is DH with a Twist

1. Alice and Bob do Diffie Helman over mod p. Let $n = \lceil \lg p \rceil$. All elements of $\mathbb{Z}_p$ are n-bit strings.
2. Alice and Bob share secret $s = g^{ab}$. View as a bit string.
3. To send m, Alice sends $c = m \oplus s$
4. To decrypt, Bob computes $c \oplus s = c \oplus s \oplus s = m \pmod{p}$

Why is ElGamal used and ElGasarch is not? Discuss

ElGasarch is DH with a Twist

1. Alice and Bob do Diffie Helman over mod p. Let $n = \lceil \lg p \rceil$. All elements of $\mathbb{Z}_p$ are n-bit strings.
2. Alice and Bob share secret $s = g^{ab}$. View as a bit string.
3. To send m, Alice sends $c = m \oplus s$
4. To decrypt, Bob computes $c \oplus s = c \oplus s \oplus s = m \pmod{p}$

Why is ElGamal used and ElGasarch is not? Discuss
Example: $p = 23$. The elements are $\{0, \ldots, 22\}$. $0, \ldots, 15$ use 4 bits. $16, \ldots, 22$ use 5 bits. So if all use 5 bits then $15/22 \sim 0.68$ of the strings have a 0 as first bit. Not Random Enough.

Could ElGasarch work with some variant of DH? Discuss

ElGasarch is DH with a Twist

1. Alice and Bob do Diffie Helman over mod p. Let $n = \lceil \lg p \rceil$. All elements of $\mathbb{Z}_p$ are n-bit strings.
2. Alice and Bob share secret $s = g^{ab}$. View as a bit string.
3. To send m, Alice sends $c = m \oplus s$
4. To decrypt, Bob computes $c \oplus s = c \oplus s \oplus s = m \pmod{p}$

Why is ElGamal used and ElGasarch is not? Discuss

Example: $p = 23$. The elements are $\{0, \ldots, 22\}$. $0, \ldots, 15$ use 4 bits. $16, \ldots, 22$ use 5 bits. So if all use 5 bits then $15/22 \sim 0.68$ of the strings have a 0 as first bit. Not Random Enough.

Could ElGasarch work with some variant of DH? Discuss

Would need to do DH over a group (1) with power-of-2 elts, (2) DL is hard, (3) mult is easy. None exist (yet).

Public Key Cryptography: RSA

Needed Mathematics- The ϕ Function

Known: If p is prime then $a^{p-1} \equiv 1 \pmod{p}$.
Ramifications: For all m, $a^m \equiv a^{m \pmod{p-1}} \pmod{p}$.
So arithmetic in the exponents is mod $p-1$.

We need to generalize this.

Definition

$\phi(n)$ is the number of numbers in $\{1, \ldots, n-1\}$ that are relatively prime to n.

Note: If p is prime then $\phi(p) = p - 1$.
Known: If n is any number then $a^{\phi(n)} \equiv 1 \pmod{n}$.
Ramifications: For all m, $a^m \equiv a^{m \pmod{\phi(n)}} \pmod{n}$.

Needed Mathematics- Examples

$14^{400} \pmod{1009}$. Repeated squaring takes

$$\lceil \lg(400) \rceil = 9 \text{ steps}$$

$14^{4,000,000,000} \pmod{1009}$. Repeated squaring takes

$$\lceil \lg(4,000,000,000) \rceil = 32 \text{ steps}$$

Can we do better?
$\phi(1009) = 1008$.
$4,000,000,000 \equiv 976 \pmod{1008}$

$$14^{4,000,000,000} \equiv 14^{976} \pmod{1009}$$

Now do repeated squaring which take

$$\lceil \lg(976 \rceil = 10 \text{ steps}$$

More Needed Mathematics

Known: If a, b are relatively prime then $\phi(ab) = \phi(a)\phi(b)$.

Known: Given R, easy to find e rel prime to R and d such that $ed \equiv 1 \pmod{R}$.

Believe: Let $N = pq$, $R = (p-1)(q-1)$ and e rel prime to R.
If know N but Not R then hard to find d with $ed \equiv 1 \pmod{R}$.

RSA

Let n be a security parameter

1. Alice picks two primes p, q of length n and computes $N = pq$.
2. Alice computes $\phi(N) = \phi(pq) = (p-1)(q-1)$. Denote by R
3. Alice picks an $e \in \{\frac{R}{3}, \ldots, \frac{2R}{3}\}$ that is relatively prime to R. Alice finds d such that $ed \equiv 1 \pmod{R}$.
4. Alice broadcasts (N, e). (Bob and Eve both see it.)
5. Bob: To send message $m \in \{1, \ldots, N-1\}$, send $m^e \pmod{N}$.
6. If Alice gets $m^e \pmod{N}$ she computes

$$(m^e)^d \equiv m^{ed} \equiv m^{ed \pmod{R}} \equiv m^{1 \pmod{R}} \equiv m$$

RSA

Let n be a security parameter

1. Alice picks two primes p, q of length n and computes $N = pq$.
2. Alice computes $\phi(N) = \phi(pq) = (p-1)(q-1)$. Denote by R
3. Alice picks an $e \in \{\frac{R}{3}, \ldots, \frac{2R}{3}\}$ that is relatively prime to R. Alice finds d such that $ed \equiv 1 \pmod{R}$.
4. Alice broadcasts (N, e). (Bob and Eve both see it.)
5. Bob: To send message $m \in \{1, \ldots, N-1\}$, send $m^e \pmod{N}$.
6. If Alice gets $m^e \pmod{N}$ she computes

$$(m^e)^d \equiv m^{ed} \equiv m^{ed \pmod{R}} \equiv m^{1 \pmod{R}} \equiv m$$

PRO: Alice and Bob can execute the protocol easily.

RSA

Let n be a security parameter

1. Alice picks two primes p, q of length n and computes $N = pq$.
2. Alice computes $\phi(N) = \phi(pq) = (p-1)(q-1)$. Denote by R
3. Alice picks an $e \in \{\frac{R}{3}, \ldots, \frac{2R}{3}\}$ that is relatively prime to R. Alice finds d such that $ed \equiv 1 \pmod{R}$.
4. Alice broadcasts (N, e). (Bob and Eve both see it.)
5. Bob: To send message $m \in \{1, \ldots, N-1\}$, send $m^e \pmod{N}$.
6. If Alice gets $m^e \pmod{N}$ she computes

$$(m^e)^d \equiv m^{ed} \equiv m^{ed \pmod{R}} \equiv m^{1 \pmod{R}} \equiv m$$

PRO: Alice and Bob can execute the protocol easily.
Biggest PRO: Alice and Bob never had to meet!

RSA

Let n be a security parameter

1. Alice picks two primes p, q of length n and computes $N = pq$.
2. Alice computes $\phi(N) = \phi(pq) = (p-1)(q-1)$. Denote by R
3. Alice picks an $e \in \{\frac{R}{3}, \ldots, \frac{2R}{3}\}$ that is relatively prime to R. Alice finds d such that $ed \equiv 1 \pmod{R}$.
4. Alice broadcasts (N, e). (Bob and Eve both see it.)
5. Bob: To send message $m \in \{1, \ldots, N-1\}$, send $m^e \pmod{N}$.
6. If Alice gets $m^e \pmod{N}$ she computes

$$(m^e)^d \equiv m^{ed} \equiv m^{ed \pmod{R}} \equiv m^{1 \pmod{R}} \equiv m$$

PRO: Alice and Bob can execute the protocol easily.
Biggest PRO: Alice and Bob never had to meet!
Question: Can Eve find out m?

Do RSA in Class

Pick out two students to be Alice and Bob.
Use primes
$p = 31$, Prime
$q = 37$, Prime
$N = pq = 31 * 37 = 1147$.
$R = \phi(N) = 30 * 36 = 1080$
$e = 77$ (e rel prime to R)
$d = 533$ ($ed \equiv 1 \pmod{R}$)
CHECK: $ed = 77 * 533 = 41041 \equiv 1 \pmod{1080}$.
Bob: pick an $m \in \{1, \ldots, N-1\} = \{1, \ldots, 1147\}$. Do not tell us what it is.
Bob: compute $c = m^e \pmod{1147}$ and tell it to us.
Alice: compute $c^d \pmod{1147}$, should get back m.

What Do We Really Know about RSA

If Eve can factor then she can crack RSA.

1. Input (N, e) where $N = pq$ and e is rel prime to $R = (p-1)(q-1)$. $(p, q, R$ are NOT part of the input.)
2. Eve factors N to find p, q. Eve computes $R = (p-1)(q-1)$.
3. Eve finds d such that $ed \equiv 1 \pmod{R}$.

Converse is not known to be true!

Definition: Let f be the following function:
Input: N, e where $N = pq$ and e is rel prime to $R = (p-1)(q-1)$ $(p, q$ are NOT in the input).
Output: d such that $ed \equiv 1 \pmod{R}$.

Hardness assumption (HA): f is hard to compute.
VOTE: HA implies RSA secure? YES, NO, UNKNOWN

What Do We Really Know about RSA

If Eve can factor then she can crack RSA.

1. Input (N, e) where $N = pq$ and e is rel prime to $R = (p-1)(q-1)$. $(p, q, R$ are NOT part of the input.)
2. Eve factors N to find p, q. Eve computes $R = (p-1)(q-1)$.
3. Eve finds d such that $ed \equiv 1 \pmod{R}$.

Converse is not known to be true!

Definition: Let f be the following function:
Input: N, e where $N = pq$ and e is rel prime to $R = (p-1)(q-1)$ (p, q are NOT in the input).
Output: d such that $ed \equiv 1 \pmod{R}$.

Hardness assumption (HA): f is hard to compute.
VOTE: HA implies RSA secure? YES, NO, UNKNOWN
NO

Plain RSA Bytes!

The RSA given above is referred to as Plain RSA.
Insecure!

Plain RSA Bytes!

The RSA given above is referred to as Plain RSA.
Insecure!

Scenario:
Eve sees Alice send Bob c_1.

Plain RSA Bytes!

The RSA given above is referred to as Plain RSA.
Insecure!

Scenario:
Eve sees Alice send Bob c_1.
Later Eve sees Alice send Bob c_2.

Plain RSA Bytes!

The RSA given above is referred to as Plain RSA.
Insecure!

Scenario:
Eve sees Alice send Bob c_1.
Later Eve sees Alice send Bob c_2.

What can Eve easily deduce?

Plain RSA Bytes!

The RSA given above is referred to as Plain RSA.
Insecure!

Scenario:
Eve sees Alice send Bob c_1.
Later Eve sees Alice send Bob c_2.

What can Eve easily deduce?

Eve can know if $c_1 = c_2$ or not.

That alone makes it insecure.

Plain RSA Bytes!

The RSA given above is referred to as Plain RSA.
Insecure!

Scenario:
Eve sees Alice send Bob c_1.
Later Eve sees Alice send Bob c_2.

What can Eve easily deduce?

Eve can know if $c_1 = c_2$ or not.

That alone makes it insecure.
Plain RSA is never used and should never be used!

PKCS-1.5 RSA

How can we fix RSA to make it work? Discuss

PKCS-1.5 RSA

How can we fix RSA to make it work? Discuss Need randomness.

PKCS-1.5 RSA

How can we fix RSA to make it work? Discuss Need randomness.

We need to change how Bob sends a message;
BAD: To send $m \in \{1, \ldots, N-1\}$, send $m^e \pmod{N}$.

GOOD?: To send $m \in \{1, \ldots, N-1\}$, pick rand r, send $(rm)^e$.
(NOTE- rm means r CONCAT with m here and elsewhere.)

PKCS-1.5 RSA

How can we fix RSA to make it work? Discuss Need randomness.

We need to change how Bob sends a message;
BAD: To send $m \in \{1, \ldots, N-1\}$, send $m^e \pmod{N}$.

GOOD?: To send $m \in \{1, \ldots, N-1\}$, pick rand r, send $(rm)^e$.
(NOTE- rm means r CONCAT with m here and elsewhere.)

DEC: Alice can find rm but doesn't know divider.

PKCS-1.5 RSA

How can we fix RSA to make it work? Discuss Need randomness.

We need to change how Bob sends a message;
BAD: To send $m \in \{1, \ldots, N-1\}$, send $m^e \pmod{N}$.

GOOD?: To send $m \in \{1, \ldots, N-1\}$, pick rand r, send $(rm)^e$.
(NOTE- rm means r CONCAT with m here and elsewhere.)

DEC: Alice can find rm but doesn't know divider.
Still bytes! How to fix? Discuss.

PKCS-1.5 RSA

How can we fix RSA to make it work? Discuss Need randomness.

We need to change how Bob sends a message;
BAD: To send $m \in \{1, \ldots, N-1\}$, send $m^e \pmod N$.

GOOD?: To send $m \in \{1, \ldots, N-1\}$, pick rand r, send $(rm)^e$.
(NOTE- rm means r CONCAT with m here and elsewhere.)

DEC: Alice can find rm but doesn't know divider.
Still bytes! How to fix? Discuss.

Let $L_1 = \left\lfloor \frac{\lg N}{3} \right\rfloor$, $L_2 = \lfloor \lg N \rfloor - L$.
To send $m \in \{0,1\}^{L_2}$ pick random $r \in \{0,1\}^{L_1}$.
When Alice gets rm she will know that m is the last L_2 bits.

Is PKCS-1.5 RSA Secure?

Is PKCS-1.5 RSA Secure? VOTE

- YES (under hardness assumptions and large n)
- NO (there is yet another weird security thing we overlooked)

Is PKCS-1.5 RSA Secure?

Is PKCS-1.5 RSA Secure? VOTE

- YES (under hardness assumptions and large n)
- NO (there is yet another weird security thing we overlooked)

NO (there is yet another weird security thing we overlooked)

Is PKCS-1.5 RSA Secure?

Is PKCS-1.5 RSA Secure? VOTE

- YES (under hardness assumptions and large n)
- NO (there is yet another weird security thing we overlooked)

NO (there is yet another weird security thing we overlooked)
Scenario: N and e are public. Bob sends $(rm)^e \pmod{N}$.
Eve cannot determine what m is.

Is PKCS-1.5 RSA Secure?

Is PKCS-1.5 RSA Secure? VOTE

- YES (under hardness assumptions and large n)
- NO (there is yet another weird security thing we overlooked)

NO (there is yet another weird security thing we overlooked)
Scenario: N and e are public. Bob sends $(rm)^e \pmod{N}$.
Eve cannot determine what m is.
What can Eve do that is still obnoxious?

Is PKCS-1.5 RSA Secure?

Is PKCS-1.5 RSA Secure? VOTE

- YES (under hardness assumptions and large n)
- NO (there is yet another weird security thing we overlooked)

NO (there is yet another weird security thing we overlooked)
Scenario: N and e are public. Bob sends $(rm)^e \pmod{N}$.
Eve cannot determine what m is.
What can Eve do that is still obnoxious?
Eve can compute $2^e(rm)^e \equiv (2(rm))^e \pmod{N}$. So what?

Is PKCS-1.5 RSA Secure?

Is PKCS-1.5 RSA Secure? VOTE

- YES (under hardness assumptions and large n)
- NO (there is yet another weird security thing we overlooked)

NO (there is yet another weird security thing we overlooked)
Scenario: N and e are public. Bob sends $(rm)^e \pmod{N}$.
Eve cannot determine what m is.
What can Eve do that is still obnoxious?
Eve can compute $2^e(rm)^e \equiv (2(rm))^e \pmod{N}$. So what?

Eve can later pretend she is Bob and send $(2(rm))^e \pmod{N}$.

Is PKCS-1.5 RSA Secure?

Is PKCS-1.5 RSA Secure? VOTE

- YES (under hardness assumptions and large n)
- NO (there is yet another weird security thing we overlooked)

NO (there is yet another weird security thing we overlooked)
Scenario: N and e are public. Bob sends $(rm)^e \pmod{N}$.
Eve cannot determine what m is.
What can Eve do that is still obnoxious?
Eve can compute $2^e(rm)^e \equiv (2(rm))^e \pmod{N}$. So what?

Eve can later pretend she is Bob and send $(2(rm))^e \pmod{N}$.

Why bad? Discuss

Is PKCS-1.5 RSA Secure?

Is PKCS-1.5 RSA Secure? VOTE

- YES (under hardness assumptions and large n)
- NO (there is yet another weird security thing we overlooked)

NO (there is yet another weird security thing we overlooked)
Scenario: N and e are public. Bob sends $(rm)^e \pmod{N}$.
Eve cannot determine what m is.
What can Eve do that is still obnoxious?
Eve can compute $2^e(rm)^e \equiv (2(rm))^e \pmod{N}$. So what?

Eve can later pretend she is Bob and send $(2(rm))^e \pmod{N}$.

Why bad? Discuss
(1) will confuse Alice (2) Sealed Bid Scenario.

Malleability

An encryption system is malleable if when Eve sees a message she can figure out a way to send a similar one, where she knows the similarity (she still does not know the message).

1. The definition above is informal.
2. Can modify RSA so that its probably not malleable.
3. That way is called PKCS-2.0-RSA.
4. Name BLAH-1.5 is hint that its not final version.
5. Will study PKCS-2.0.RSA later in the course.

Final Points About RSA

1. PKCS-2.0-RSA is REALLY used!
2. There are many variants of RSA but all use the ideas above.
3. We may show (much) later show how to prove, assuming the hardness assumption, that RSA is hard to crack.
4. Factoring easy implies RSA crackable. TRUE.
5. RSA crackable implies Factoring easy: UNKNOWN.
6. RSA crackable implies Factoring easy: Often stated in expositions of crypto. They are wrong!
7. Timing attacks on RSA bypass the math.