50.020 Security
Lecture 15 - Public Key Crypto

Recap

Last Lecture:

- We stared Public-Key cryptography
- Focused on Diffie-Hellman Key Exchange (DHKE)
  - How DHKE works
  - Why DHKE is secure? Solving Discrete-Logarithm Problem (which is hard for well-chosen groups)
  - How modular exponentiation works (needed at both Sender and Receiver)

A bit more about DLP: DLP solving records

- As DLP hardness is just assumed, practical tests are needed
- For that reason, practical DLP solving competitions exist
  - Different categories: mod p, Galois fields, elliptic curves
- Recommendation of constructing secure systems based on DLP: prime number p should have length of at least 2048 bits
- Elliptic-curve cryptography (ECC) seems to still be secure for smaller primes

This lecture

- This lecture: More examples on public key crypto
  - Elgamal
  - RSA
- Some content in these slides based on slide set of "Understanding Cryptography" by C. Paar and J. Petzl

Introduction

- The prime advantage of public-key cryptography is increased security - the private keys do not ever need to be transmitted or revealed to anyone
- We learned about Diffie-Hellman key exchange (DHKE), how does it relate to public key encryption (not just for establishing a key)?
- Lets assume we can multiply and exponentiate mod p
  - Can we build an encryption scheme on top of that?
  - The scheme should encrypt directly, and not just establish a key for AES or similar

Elgamal

Elgamal

- Elgamal is similar to DHKE, but includes encryption of message

has m

selects p, g, b

$B = g^b \bmod p$

p, g, B ⟵

selects a

computes $A = g^a$ mod p

computes $c = mB^a$ mod p

A, c ⟶

$B^a = A^b$ mod p

$m = c/B^a$ mod p

RSA

RSA

- Introduction
- Construction
  - How It works
  - Intuition for parameters in RSA (Pg 13-Pg16, not required for this course)
  - Why It Is Secure
- How to Use RSA
  - Encrypted key exchange
  - Digital signature (next week)

RSA: Introduction

- First published in 1977, invented by Ron Rivest, Adi Shamir, and Leonard Adleman.
- Currently commonly applied into Internet security. Examples of application:
  - Most Public Key Infrastructure (PKI) products.
  - SSL/TLS: Certificates and key-exchange
  - Secure e-mail: Outlook, Pretty Good Privacy (PGP), etc.

RSA Construction

RSA: How It Works

selects large primes p, q

$n = pq, \phi(n) = (p-1)(q-1)$

selects e, computes d

m

$$\xrightarrow{e, n}$$

$c = m^e \bmod n$

$$\xleftarrow{c}$$

$$m = c^d = m^{ed} = m^{\phi(n)+1} \bmod n$$

- e is chosen such that $1 < e < \phi(n)$ and gcd(e,ϕ(n))=1
- d is computed as the inverse of e mod ϕ(n)

RSA key pair:

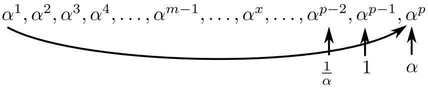
- Public key: e,n
- Private key: d

Intuition for parameters in RSA

To understand the intuition for parameters in RSA, we need to understand Fermat's Little Theorem and Euler's Totient Function (in the next two slides).

Intuition for parameters in RSA: Fermat's little theorem

- For prime p, with any $a < p$
  - $a^p \equiv a \bmod p$
- This is also related to our generating elements in prime fields



Visualization of $\alpha^{-1} \equiv \alpha^{p-2}$

Intuition for parameters in RSA: Euler's totient function

- Generalization for groups with non-prime modulus
- $\alpha^{\phi(n)} \equiv 1 \bmod n$
- Here, $\phi(n)$ is *Euler's totient* function
- $\phi(n)$ gives the number of elements that are co-prime to n (and are smaller than n)
  - Co-prime means that both number's gcd is 1
  - If n is prime, $\phi(n)$ is n-1
  - Example: $\phi(6)=2$, as 1,5 are co-prime to 6
- For a product mn, we have $\phi(mn)=\phi(m)*\phi(n)$
  - If m,n are relatively prime
  - So $\phi(2*3)=\phi(2)*\phi(3)=1*2$

Intuition for parameters in RSA

- In RSA, d,e and modulus n are chosen in a specific way
- Using Fermat's little Theorem and Euler's Totient
- We know that $\mathrm{m}^{\mathrm{de}} \equiv$ m mod n iff de is prime
    - But a product cannot be prime...
- We choose n as product of two large primes p,q
- We also know that $\mathrm{m}^{\phi(\mathrm{n})} \equiv 1$ mod n
    - So $\mathrm{m}^{\phi(\mathrm{n})+1} \equiv$ m mod n
    - So we have to find de$=\phi$(n)+1
    - Or any de that satisfies de$=$ kϕ(n)+1 (for some integer k)
        - Because $m^{\phi(n)^k} \equiv 1$ mod n
        - We can also write de$\equiv\phi$(n)+1 mod ϕ(n)
    - Choose $1 < e < \phi(n)$ and gcd(e,ϕ(n))=1 as public key
    - Compute d as the inverse of e mod ϕ(n)

RSA: Why It Is Secure

RSA: Why It Is Secure

- Big integer factorization problem: decomposition of a composite number into a product of smaller integers. If these integers are further restricted to prime numbers, the process is called prime factorization.
- For RSA:
  - Given big $n = pq$ (n is recommended to be 2048 bits), it is hard to find the two prime numbers p and q.
  - Hard to find p and q, and thus hard to retrieve $\phi(n) = (p-1)(q-1)$ which is used to compute the key pair (including the private key).

RSA: How To Use It?

How to use RSA

- Using RSA, we can do the following
  - Encryption using $m^e \bmod n$
  - Decryption using $m^d \bmod n$
- Only the holder of d can decrypt, everyone else can encrypt
- But both operation are slow
  - Can be 1000 times slower than AES
- *RSA is practically never used to encrypt messages*
- How could we use RSA instead?

How to use RSA

- Using RSA, we can do the following
  - Encryption using $m^e \bmod n$
  - Decryption using $m^d \bmod n$
- Only the holder of d can decrypt, everyone else can encrypt
- But both operation are slow
  - Can be 1000 times slower than AES
- *RSA is practically never used to encrypt messages*
- How could we use RSA instead?

- Encrypted key exchange
- *Digital Signatures* (next week)

Encrypted key exchange

- Alice wants to send a lot of data to Bob
- They can use RSA and AES, Bob shared a public key e_B, n_B
- Alice generates a k_{AB}, and computes $c = k_{AB}^{e_B} \bmod n_B$
- Alice then encrypts the data with AES, using k_{AB} as key
- Alice sends c to Bob, and Bob recovers k_{AB} by computing $c^{d_B} \bmod n_B$
- Alice sends encrypted data to Bob, and Bob used AES and k_{AB} to decrypt it

Malleability of RSA

- We saw earlier that symmetric encryption does not always protect integrity
  - Attack on stream ciphers demonstrated by us
  - Changed "buy100" to "buy999"
- (Textbook) RSA encryption is also malleable, in a similar way
- In particular, if m="100", and c=m^e mod n, then:
  - for m'=200 c'=2^e*c mod n
- So the attacker can change the message deterministically
  - without knowing it
- In addition, RSA encryption is deterministic (like ECB-AES)
- How to protect against this? Apply some (reversible) transformation before encryption
  - if this introduces randomness, we break encryption determinism
  - OAEP is such a scheme

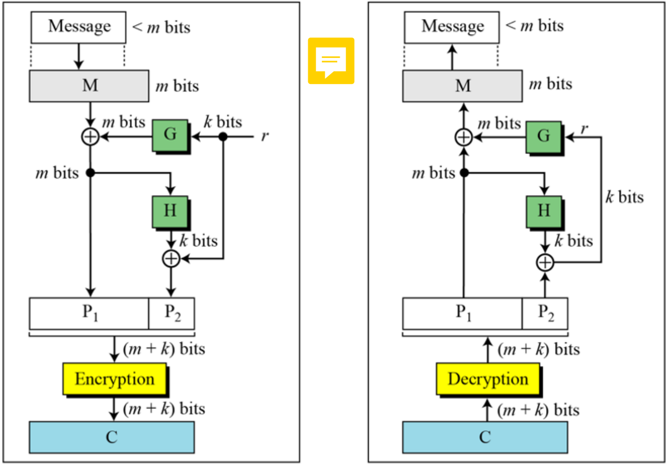
Optimal asymmetric encryption padding

Notes: G and H are typically some cryptographic hash functions

M: Padded message
r: One-time random number

P: Plaintext ($P_1 \| P_2$)
C: Ciphertext

G: Public function (k-bit to m-bit)
H: Public function (m-bit to k-bit)



Optimal asymmetric encryption padding

- OAEP is a stronger padding scheme that is often used instead of just a hash
- OAEP uses a Feistel network to transform message before the signature
- Core is the application of XORs and two hash functions
- Uses nonce, so is gives non-deterministic result
- Nonce does not have to sent explicitly, can be recovered
- Proven secure under the RSA assumption when G and H are secure.

"Decrypting Plaintext"

- So far, Bob used e to encrypt data (to obtain ciphertext), and Alice used d to decrypt the ciphertext to data
- Only Alice's d can decrypt the ciphertext
- What happens if Alice uses d to decrypt *plaintext*?
  - Who can do what with the result?

"Decrypting Plaintext"

- So far, Bob used e to encrypt data (to obtain ciphertext), and Alice used d to decrypt the ciphertext to data
- Only Alice's d can decrypt the ciphertext
- What happens if Alice uses d to decrypt *plaintext*?
  - Who can do what with the result?

- The public key can reverse the previous action
- This will result in the original text

"Decrypting Plaintext"

- So far, Bob used e to encrypt data (to obtain ciphertext), and Alice used d to decrypt the ciphertext to data
- Only Alice's d can decrypt the ciphertext
- What happens if Alice uses d to decrypt *plaintext*?
  - Who can do what with the result?

- The public key can reverse the previous action
- This will result in the original text

- This can be used to verify that the original text came from Alice
- The "original text" will be a hash value of some message

Conclusion

- We introduced public key establishement last lecture (DHKE).
- Elgamal and RSA are two public key crypto examples to encry/decrypt messages.
- RSA is still widely used
  - But almost never to encrypt messages/data.
  - Main use: key exchange/establishment, signatures.
  - RSA encryption is also malleable; optimal asymmetric encryption padding can be used together with RSA to enhance RSA's security.