# Cryptographic Primitives II

Brad Karp
UCL Computer Science

CS GZ03 / M030
24th November, 2010

# Misuses of RSA Break Secrecy

- When encrypting, what if plaintext drawn from very small set (e.g., {"yes", "no"})?
- Employees escrow secret documents, encrypted with company's public key
  - Upon firing or death of one employee, company releases plaintext to another
  - Employee E takes employee A's ciphertext $c = m^e \bmod n$, escrows $c2^e \bmod n$
  - Employee E fired; co-conspirator F gets 2m!
- **Chosen ciphertext attack (CCA):** eavesdrop a ciphertext c; submit specially concocted messages for decryption; study resulting plaintexts; learn plaintext, $m = c^d \bmod n$

# RSA: Not Quite Exponentiation

- At first glance, RSA operations appear to be raising a message to a power
- But they're not, really…the mod n means RSA in fact a trap-door permutation
  - Map one element, m, of set {0, …, n-1} to another, c
  - Not invertible without knowing d
- Non-invertibility applies to whole of m and c; not to individual bits of m and c, or other properties over m and c, e.g., parity of m
  - In escrow attack, multiplicative relationship among RSA ciphertexts exists, despite non-invertibility
- It's possible that learning even one bit of m may help recover all of m from c

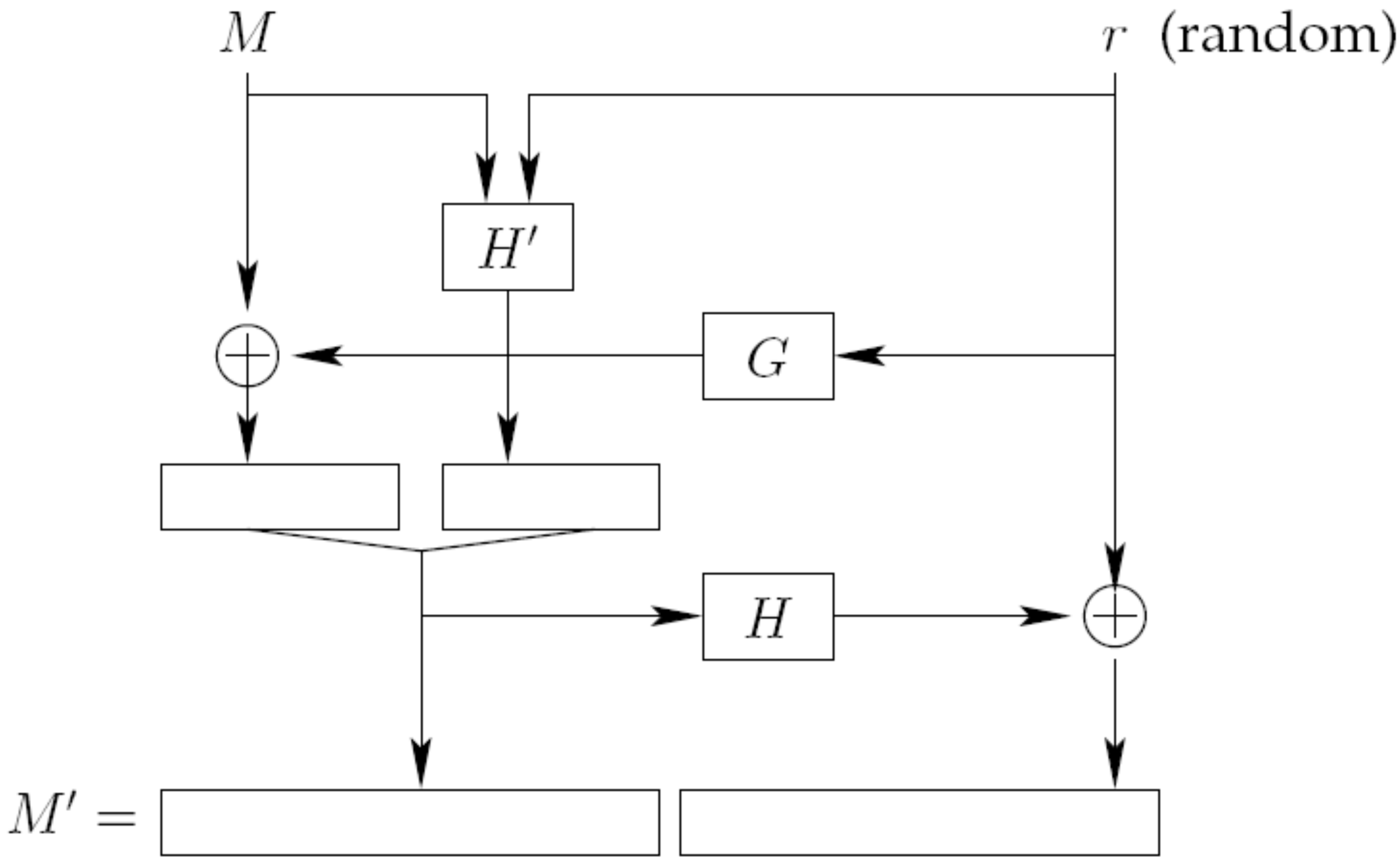
# Adaptive Chosen Ciphertext Attack on RSA in SSL 3.0

- SSL 3.0 encrypted with RSA by padding plaintext into blocks using PKCS #1 standard, as follows:
  - 0x00 | 0x02 |
    8 or more non-zero random bytes | 0x00 |
    plaintext block
- SSL decrypts received ciphertext, checks if result in this format; returns “format error” if not!
- Bleichenbacher’s adaptive CCA attack: with about one million messages to server, attacker can recover m for previously eavesdropped ciphertext $c = m^e \bmod n$
  - When chosen ciphertext accepted by server, attacker knows first two plaintext bytes with certainty!

# Making RSA Secure Against Adaptive CCA Attacks

- Intuition: want plaintext input to RSA to be all-or-nothing transform of actual message
  - e.g., so that multiplicative property over ciphertexts doesn't reveal message, and knowing one bit doesn't reveal anything about whole message
- Desirable transform properties:
  - Randomness: unique ciphertext for repeated identical messages
  - Redundancy: make most strings invalid ciphertexts
  - Entanglement: knowing partial information about input to RSA should reveal nothing about message
  - Invertibility: of course, must be able to recover original message when decrypting

# Practical Padding for RSA: OAEP+ [Shoup]




- Transforms message M into RSA input M'
- Not proven adaptive CCA secure, but heuristically so

# Digital Signatures with RSA

- RSA trap-door permutation also useful for digital signatures
- Public-key signature operations:
  - Sign: $S(K^{-1}, m) \rightarrow \{m\}_{K^{-1}}$
  - Verify: $V(K, \{m\}_{K^{-1}}, m\} \rightarrow$ {true, false}
- Provides integrity, like a MAC:
  - Cannot produce valid $<m, \{m\}_{K^{-1}}>$ pair without knowing $K^{-1}$
- With RSA:
  - Sign using private key, using trap-door applied when decrypting
  - Verify using public key, using permutation applied when encrypting

# Multiplicative Attack Against RSA Signatures

- As in CCA, attacker may try to exploit multiplicative relationship among RSA permutation inputs and outputs, to decrypt eavesdropped ciphertexts
- Eve stores ciphertext c encrypted for Alice, wants to recover corresponding m
- Using Alice's public key, {n, e}, Eve:
  - Chooses random number $r < n$
  - Computes $y = cr^e \bmod n$
  - Eve asks Alice to sign y
  - Alice sends Eve $y^d \bmod n = c^d r^{ed} \bmod n = rc^d \bmod n$
  - Eve computes $r^{-1} \bmod n$, then recovers

$$m = c^d \bmod n = r^{-1} r c^d \bmod n$$

# Multiplicative Attack Against RSA Signatures

- As in CCA, attacker may try to exploit

**Lesson:**

**Don't sign whole messages presented to you by others!**

- Eve stores ciphertext c encrypted for Alice, wants to recover corresponding m
- Using Alice's public key, {n, e}, Eve:
  - Chooses random number $r < n$
  - Computes $y = cr^e \bmod n$
  - Eve asks Alice to sign y
  - Alice sends Eve $y^d \bmod n = c^d r^{ed} \bmod n = rc^d \bmod n$
  - Eve computes $r^{-1} \bmod n$, then recovers

$$m = c^d \bmod n = r^{-1} r c^d \bmod n$$

# Only Sign Message Hashes with RSA!

- Again, want all-or-nothing transform over message before signing with trap door
- Full-domain hash:
  - Before signing message, compute hash of message sized to be same number of bits as RSA modulus n
  - Sign the hash, not the message
  - Hash reveals nothing about underlying message, nor messages arithmetically related to it

# Costs of Cryptography

- Public-key operations significantly more computationally expensive than symmetric-key ones
- Modern CPU can symmetrically encrypt and MAC faster than 100 Mbps
- Public-key encryption typically 100X slower than symmetric crypto
  - This relationship changes as hardware changes!
- Result: tend to use public-key encryption and signatures only on short messages

# Hybrid Cryptography

- Goal: mix speed of symmetric-key flexibility of public-key cryptography
- Send symmetric key encrypted with public key; message encrypted with symmetric key

# Pitfall: Public Key Provenance

- Suppose client wishes to know it's talking to particular server
- Where does client get server's public key?
- How does client know it has correct public key for real server, and not attacker?
- Man-in-the-middle attack:
  - Client connects to attacker
  - Attacker gives client attacker's public key
  - Client believes communicating with real server

# Further Reading

- The MIT Guide to Picking Locks
- Menezes, A., van Oorschot, P., and Vanstone, S., *Handbook of Applied Cryptography,* http://www.cacr.math.uwaterloo.ca/hac/
- Goldwasser, S. and Bellare, M., *Lecture Notes on Cryptography*, http://cseweb.ucsd.edu/~mihir/papers/gb.pdf
- Bleichenbacher, Daniel, Chosen Ciphertext Attacks Against Protocols Based on the RSA Encryption Standard PKCS #1, in *CRYPTO 1998*