# The RSA Cryptosystem

Dan Boneh
Stanford University

# The RSA cryptosystem

- First published:
  - Scientific American, Aug. 1977.

    (after some censorship entanglements)

- Currently the "work horse" of Internet security:
  - Most Public Key Infrastructure (PKI) products.
  - SSL/TLS: Certificates and key-exchange.
  - Secure e-mail: PGP, Outlook, …

# The RSA trapdoor permutation

- Parameters: $N = pq$. $N \approx 1024$ bits. $p,q \approx 512$ bits.
  e – encryption exponent. $\gcd(e, \varphi(N)) = 1$ .
- Permutation: $\mathbf{RSA(M) = M^e} \pmod N$ where $M \in Z_N$

---

- Trapdoor: **d** – decryption exponent.
  Where $e \cdot \mathbf{d} = 1 \pmod{\varphi(N)}$
- Inversion: $\mathbf{RSA(M)^d = M} \pmod N$

---

- "Assumption":
  no efficient alg. can invert RSA without trapdoor.

# Textbook RSA is insecure

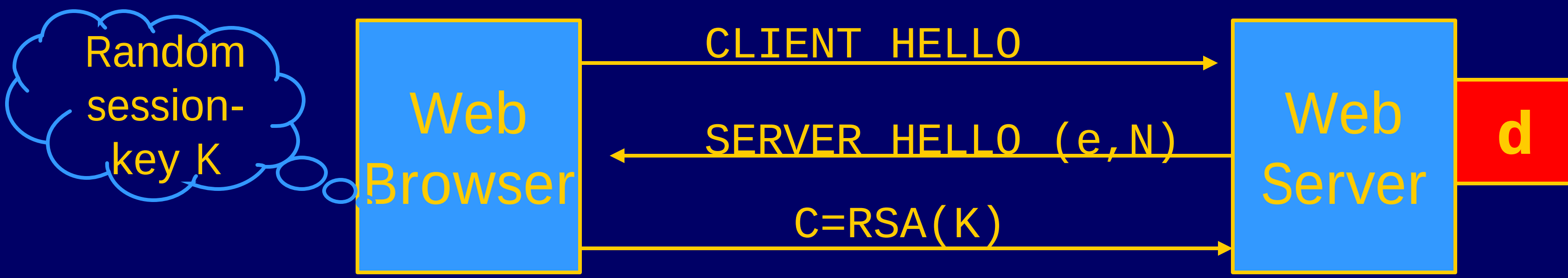
- Textbook RSA encryption:
  - public key: **(N,e)** Encrypt: $C = M^e \pmod{N}$
  - private key: **d** Decrypt: $C^d = M \pmod{N}$

    $(M \in Z_N)$

- Completely insecure cryptosystem:
  - Does not satisfy basic definitions of security.
  - Many attacks exist.

- The RSA trapdoor permutation is not a cryptosystem !

# A simple attack on textbook RSA




- Session-key K is 64 bits. View $K \in \{0,\ldots,2^{64}\}$
  Eavesdropper sees: $C = K^e \pmod{N}$ .

- Suppose $\mathbf{K = K_1 \times K_2}$ where $K_1, K_2 < 2^{34}$ . (prob. ≈20%)
  Then: $\mathbf{C/K_1^e = K_2^e} \pmod{N}$

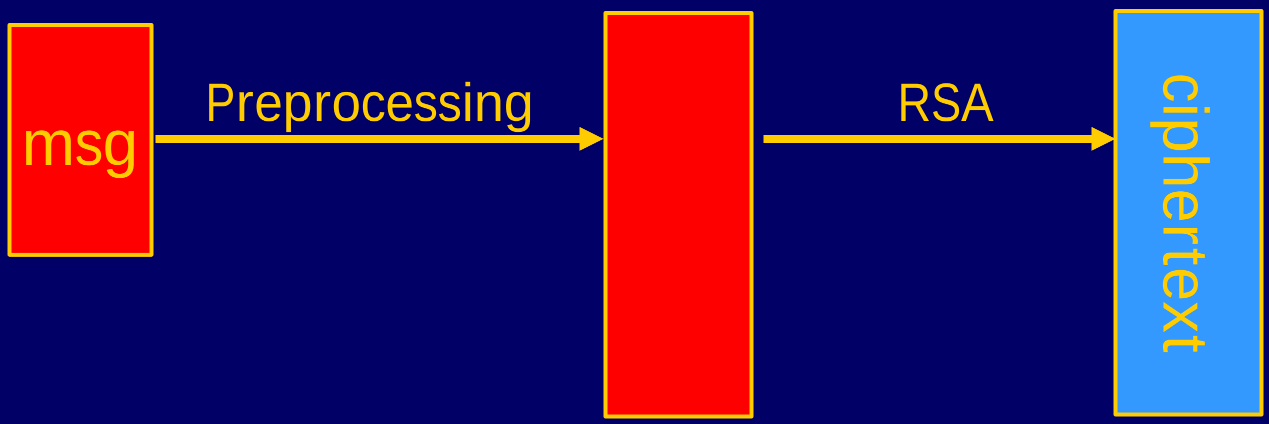
- Build table: $C/1^e, C/2^e, C/3^e, \ldots, C/2^{34e}$ . time: $2^{34}$

  For $K_2 = 0,\ldots, 2^{34}$ test if $K_2^e$ is in table. time: $2^{34} \cdot 34$

- Attack time: $\approx 2^{40} \ll 2^{64}$

# Common RSA encryption

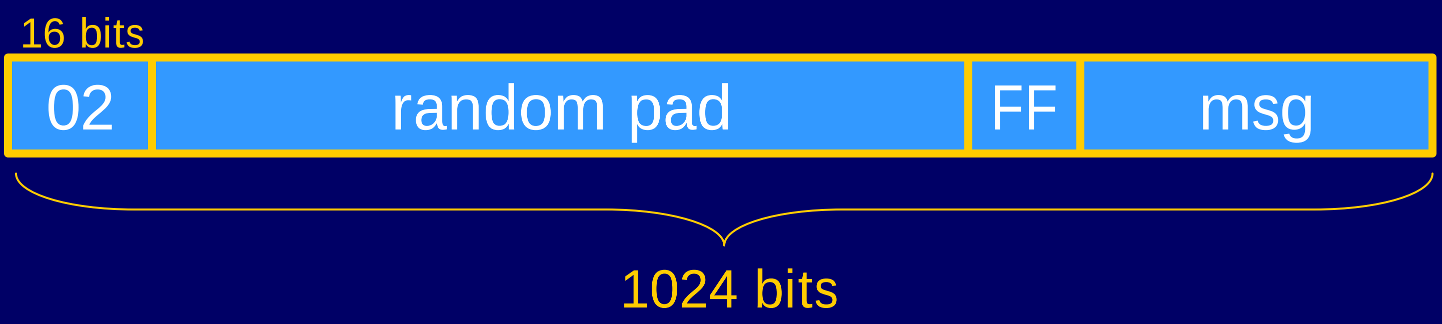
- Never use textbook RSA.
- RSA in practice:




- Main question:
  - How should the preprocessing be done?
  - Can we argue about security of resulting system?

# PKCS1 V1.5

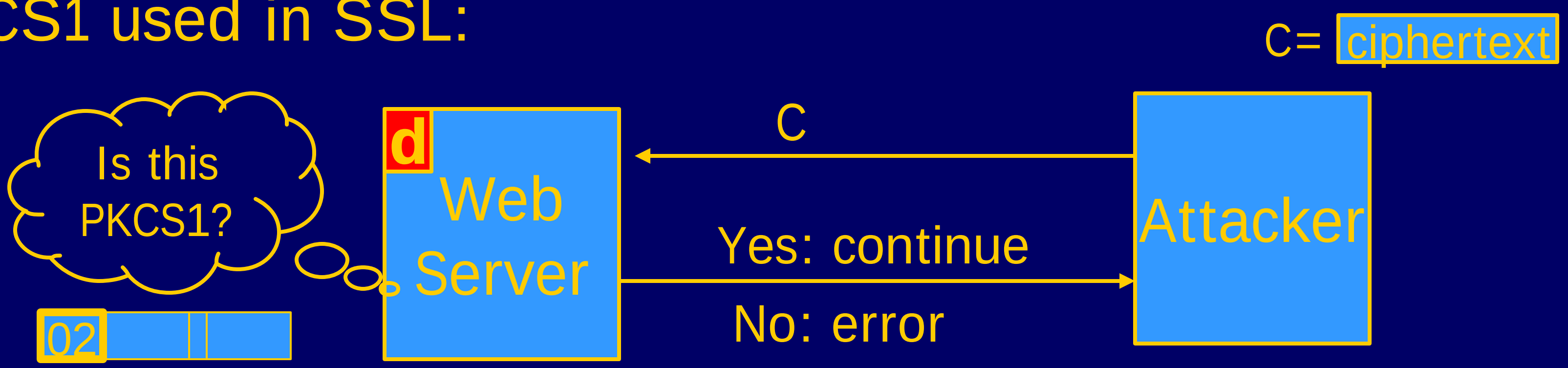
- PKCS1 mode 2: (encryption)




- Resulting value is RSA encrypted.
- Widely deployed in web servers and browsers.
- No security analysis !!

# Attack on PKCS1

- Bleichenbacher 98. Chosen-ciphertext attack.
- PKCS1 used in SSL:

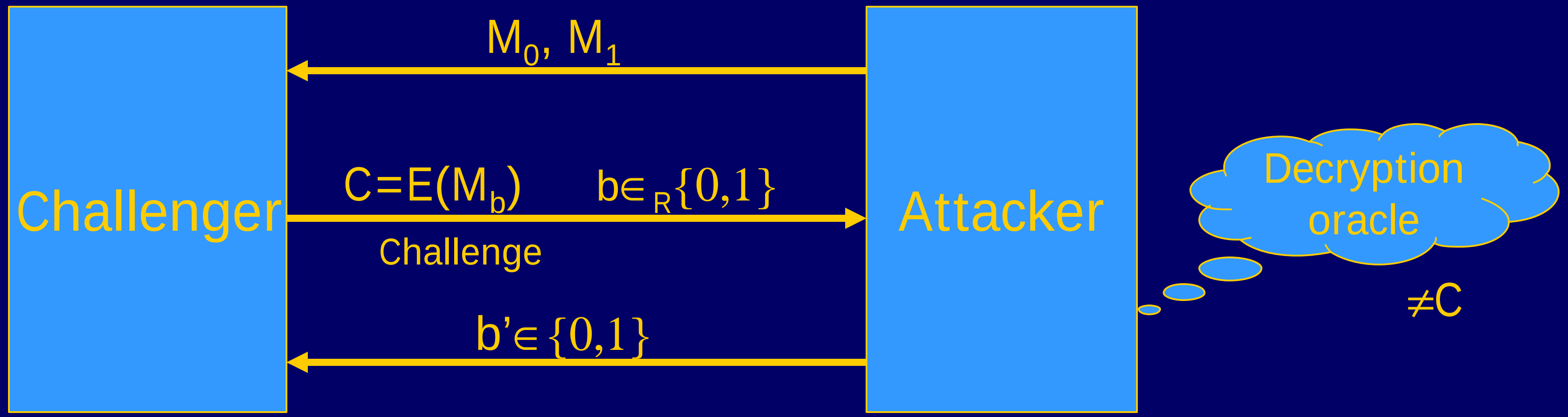



$\Rightarrow$ attacker can test if 16 MSBs of plaintext = '02'.

- Attack: to decrypt a given ciphertext C do:
  - Pick random $r \in Z_N$. Compute $C' = r^e \cdot C = (rM)^e$.
  - Send C' to web server and use response.

# Chosen ciphertext security (CCS)

- No efficient attacker can win the following game: (with non-negligible advantage)



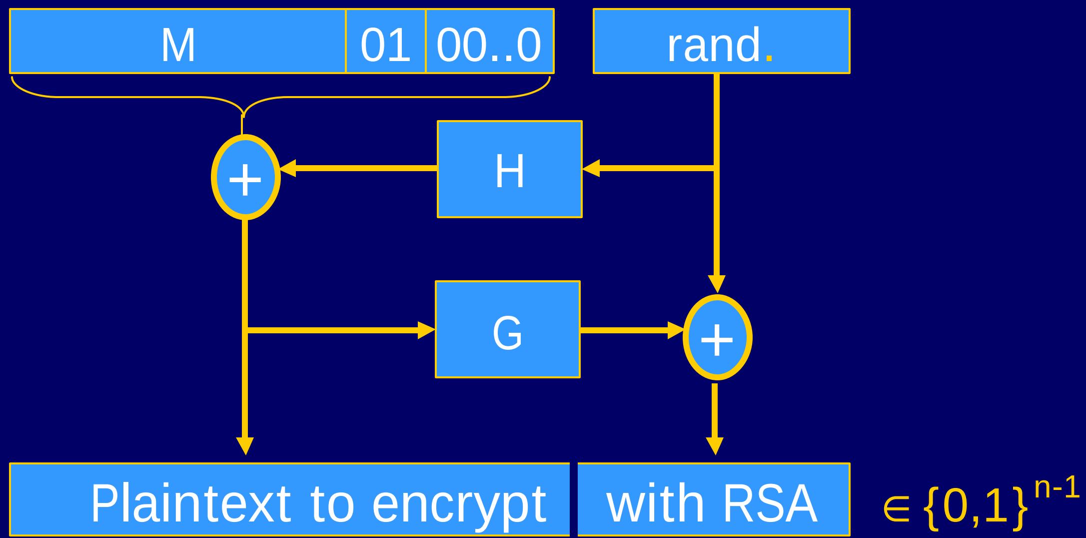

Attacker wins if $b=b'$

# Chosen-ciphertext secure RSA

- Are there CCS cryptosystems based on RSA?
  - RSA-PKCS1 is not CCS !
- Answer: Yes! Dolev-Dwork-Naor (DDN). 1991.
  - Problem: inefficient.
- Open problem: efficient CCS system based on RSA.
- What to do? Cheat!
  - Build RSA system that is CCS in imaginary world.
  - "Assume" our-world = imaginary-world.

# PKCS1 V2.0 - OAEP

- New preprocessing function: OAEP (BR94).




Check pad
on decryption.
Reject CT if invalid.

- Thm: $\forall$ trap-door permutation F $\Rightarrow$ F-OAEP is CCS
  when H,G are "*random oracles*".
- In practice: use SHA-1 or MD5 for H and G.

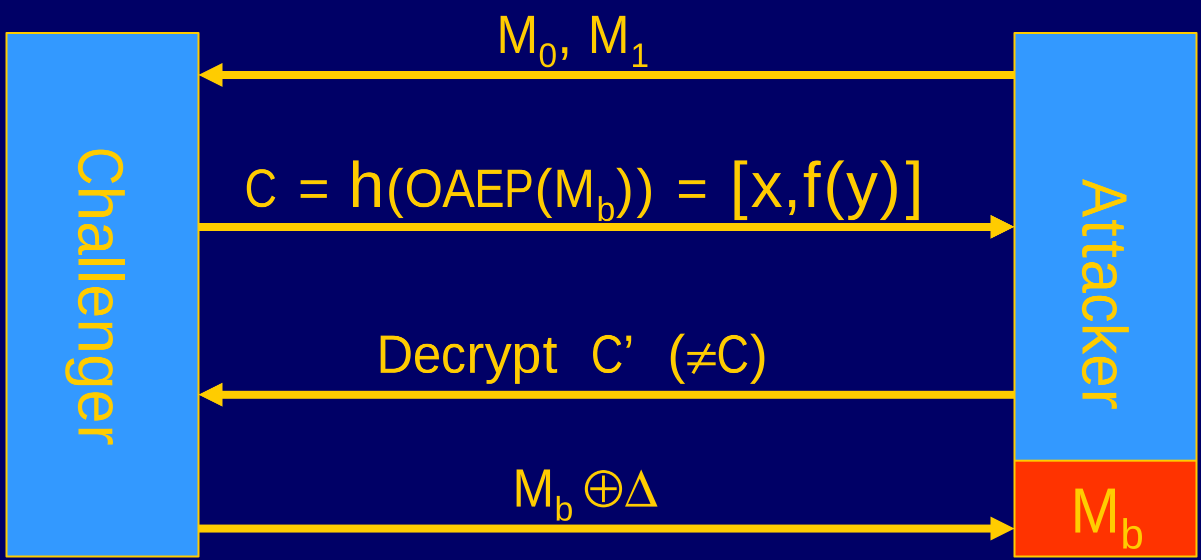
# An incorrect proof

- Shoup 2000: The OAEP thm cannot be correct !!
- Counter ex: f(x) – xor malleable trapdoor permutation

$$f(x),\ \Delta \ \Rightarrow\ f(x\oplus\Delta)$$

Define: $h(x,y) = [\ x,\ f(y)\ ]$ (also trapdoor perm)

- Attack on h-OAEP:




Rand $\Delta = r||01000$

$y' = y\oplus G(x)\oplus G(x\oplus\Delta)$

$C' = [\ x\oplus\Delta,\ f(y')\ ]$

# Consequences

- OAEP is standardized due to an incorrect thm.
- Fortunately: Fujisaki-Okamoto-Pointcheval-Stern
  - RSA-OAEP is Chosen Ciphertext Secure !!
    - Proof uses special properties of RSA.

  $\Rightarrow$ No immediate need to change standards.

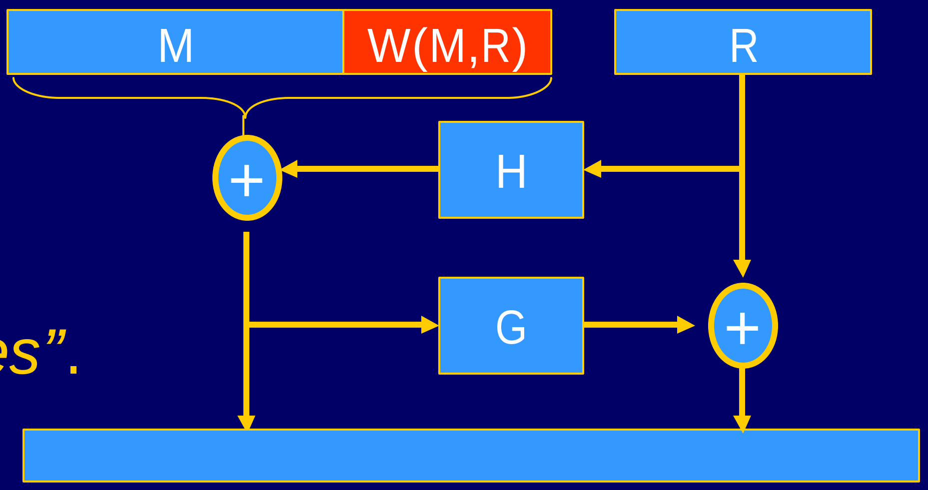
  - Security proof less efficient than original "proof".

- Main proof idea [FOPS]:
  - For Shoup's attack: given challenge C = RSA(x || y) attacker must "know" x
  - RSA(x || y) $\Rightarrow$ x then RSA is not one-way.

# OAEP Replacements

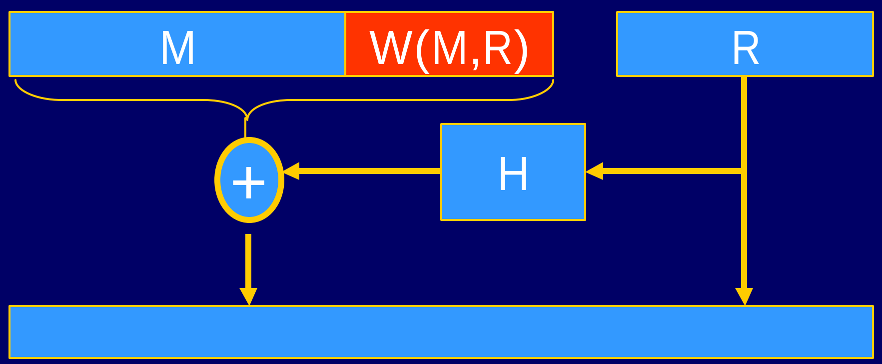
- OAEP+: (Shoup'01)

  ∀ trap-door permutation F
  F-OAEP+ is CCS when
  H,G,W are "*random oracles*".




- SAEP+: (B'01)

  RSA trap-door perm ⇒
  RSA-SAEP+ is CCS when
  H,W are "*random oracle*".



# Subtleties in implementing OAEP

```
OAEP-decrypt(C) {
    error = 0;
      ........
    if ( RSA^-1(C) > 2^(n-1) )
        { error =1;  goto exit; }
      ........
    if ( pad(OAEP^-1(RSA^-1(C))) != "01000" )
        { error = 1;  goto exit; }
}
```

- Problem: timing information leaks type of error.

  $\Rightarrow$ Attacker can decrypt any ciphertext C.

- Lesson: Don't implement RSA-OAEP yourself …

# Part II:
## Is RSA a One-Way Permutation?

# Is RSA a one-way permutation?

- To invert the RSA one-way function (without d) attacker must compute:

  M from $C = M^e \pmod{N}$.

- How hard is computing e'th roots modulo N ??
- Best known algorithm:
  - Step 1: factor N. (hard)
  - Step 2: Find e'th roots modulo p and q. (easy)

# Shortcuts?

- Must one factor N in order to compute e'th roots?
  Exists shortcut for breaking RSA without factoring?

- To prove no shortcut exists show a reduction:
  - Efficient algorithm for e'th roots mod N
    $\Rightarrow$ efficient algorithm for factoring N.
  - Oldest problem in public key cryptography.

- Evidence no reduction exists: (BV'98)
  - "Algebraic" reduction $\Rightarrow$ factoring is easy.
  - Unlike Diffie-Hellman (Maurer'94).

# Improving RSA's performance

- To speed up RSA decryption use small private key d. $C^d = M \pmod{N}$

  - Wiener87: if $d < N^{0.25}$ then RSA is insecure.
  - BD'98: if $d < N^{0.292}$ then RSA is insecure

    (open: $d < N^{0.5}$ )

  - Insecure: priv. key d can be found from (N,e).

  - Small d should never be used.

# Wiener's attack

- Recall: $e \cdot d = 1 \pmod{\varphi(N)}$

  $\Rightarrow \quad \exists\, k \in Z: \quad e \cdot d = k \cdot \varphi(N) + 1$

  $$\Rightarrow \left| \frac{e}{\varphi(N)} - \frac{k}{d} \right| \leq \frac{1}{d\varphi(N)}$$

$\varphi(N) = N-p-q+1 \quad \Rightarrow \quad |N - \varphi(N)| \leq p+q \leq 3\sqrt{N}$

$$d \leq N^{0.25}/3 \quad \Rightarrow \left| \frac{e}{N} - \frac{k}{d} \right| \leq \frac{1}{2d^2}$$

Continued fraction expansion of $e/N$ gives $k/d$.

$e \cdot d = 1 \pmod{k} \quad \Rightarrow \quad \gcd(d,k)=1$

# RSA With Low public exponent

- To speed up RSA encryption (and sig. verify) use a small e. $C = M^e \pmod N$
- Minimal value: $e=3$ ( $\gcd(e, \varphi(N)) = 1$)
- Recommended value: $e=65537=2^{16}+1$

  Encryption: 17 mod. multiplies.

- Several weak attacks. Non known on RSA-OAEP.
- Asymmetry of RSA: fast enc. / slow dec.
  - ElGamal: approx. same time for both.

# Implementation attacks

- Attack the implementation of RSA.
- Timing attack: (Kocher 97)
  The time it takes to compute $C^d \pmod N$
  can expose d.
- Power attack: (Kocher 99)
  The power consumption of a smartcard while
  it is computing $C^d \pmod N$ can expose d.
- Faults attack: (BDL 97)
  A computer error during $C^d \pmod N$
  can expose d.

  OpenSSL defense: check output. 5% slowdown.

# Key lengths

- Security of public key system should be comparable to security of block cipher.

NIST:

| Cipher key-size | Modulus size |
|---|---|
| ≤ 64 bits | 512 bits. |
| 80 bits | 1024 bits |
| 128 bits | 3072 bits. |
| 256 bits (AES) | **15360** bits |

- High security $\Rightarrow$ very large moduli.

  Not necessary with Elliptic Curve Cryptography.