601.445/645 Practical Cryptographic Systems

Asymmetric Cryptography

Instructor: Matthew Green

Housekeeping

- Programming Assignment 2 out today
  - Yeah I suck
- Weekly written assignment 2 out, due Thurs
- Syllabus/Reading:
  - Reading:
    20 years of RSA
    Imperfect Diffie Hellman
    Mining your Ps and Qs

News?

-

Diffie-Hellman Groups

- Q: How big should the modulus p be?

Diffie-Hellman Groups

- Q: How big should the modulus p be?

  A: Big enough that it's hard to solve the discrete logarithm in a reasonable time.

But what does that mean?

Solving the Discrete Log

- Let q be the order of the group
- There are various generic algorithms:
  - Brute force - O(q) group operations
  - Baby-step-giant step - time/space tradeoff m units of memory, running time O(q/m)
  - Pollard's rho - O(sqrt(q)), can be optimized to O(sqrt(n)) where n is largest prime factor of q

^ best possible approach for generic groups (Victor Shoup)

Solving the Discrete Log

- Let lql = ~1024
- Examples:
  - Brute force - ~2^{1023} (worst case)
  - Baby-step-giant step - assume 2^{32} memory
    $2^{1024} / 2^{32} = ~2^{992}$ (worst case)
  - Pollard's rho - ~2^{512}
    (assuming largest prime factor is ~1024 bits)

Solving the Discrete Log

- Let |q| = ~1024
- Examples:
  - Brute force - ~2^{1023} (worst case)
  - Baby-step-giant step - assume 2^{32} memory
    $2^{1024} / 2^{32} = ~2^{992}$ (worst case)
  - Pollard's rho - ~2^{512}
    (assuming largest prime factor is ~1024 bits)

    (# of Planck times since the Big Bang < 2^{200})

Solving the Discrete Log (2)

- But that's not the full story.
- There are various <u>special</u> algorithms that work in finite fields:
  - <u>Index calculus (NFS)</u> - $(c+o(1))(\ln n)^{1/3}(\ln \ln n)^{1-1/3})$ (c depends on nature of group)
  - Depends on the size of the <u>field</u> (not the order of the group)

Symmetric Key Size (bits)	RSA and Diffie-Hellman Key Size (bits)	Elliptic Curve Key Size (bits)
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	521

Table 1: NIST Recommended Key Sizes

Computation of a 768-bit prime field discrete logarithm

Thorsten Kleinjung[12],
Claus Diem[2], Arjen K. Lenstra[1], Christine Priplata[2], and Colin Stahlke[2]

[1] EPFL IC LACAL, Station 14, CH-1015 Lausanne, Switzerland
[2] Universität Leipzig, Mathematisches Institut, D-04009 Leipzig, Germany

Abstract. This paper reports on the number field sieve computation of a 768-bit prime field discrete logarithm, describes the different parameter optimizations and resulting algorithmic changes compared to the factorization of a 768-bit RSA modulus, and briefly discusses the cryptologic relevance of the result.
Keywords: Discrete logarithm, DSA, ElGamal, number field sieve

1 Introduction

Let $p = [2^{766}\pi] + 62762$, which is the smallest 768-bit prime number larger than $2^{766}\pi$ for which $\frac{p-1}{2}$ is prime too*. Let $g = 11$, which is a generator of the multiplicative group $\mathbf{F}_p^\times$ of the prime field $\mathbf{F}_p$. On June 16, 2016, we finished the computation of the discrete logarithm of $t = [2^{766}e]$ with respect to g. We found that the smallest non-negative integer x for which $g^x \equiv t \bmod p$ equals

$$\begin{aligned}&32592361791827056223861598597862370912834133883372105854395081352176815629509\\&16383480306379202371756381173524422992340416587484710799119774978643019959726\\&38266781162575370644813703762423329783129621567127479417280687495231463348812.\end{aligned}$$

By itself, this is a useless result. What is interesting is how we found it, that we did so with much less effort than we expected, and what the result implies for cryptographic security that relies on the difficulty of larger similar problems. These issues are discussed in this paper.

Optimization

- Making p > 1024 or 2048 bits is a drag
  - Our secret and public keys get larger
  - But more importantly: the time to compute the exponentiation g^a increases
  - Tricks:
    1. Just use short D-H exponents
    2. Find a much smaller subgroup G' inside of Z*p that we can use for this purpose, and set g to be a generator of that

    (complexity of generic DL depends on order of group, NFS etc. depend on p)

Public Key Encryption

- What if our recipient is <u>offline?</u>
  - Key agreement protocols are interac
  - e.g., want to send an email

Ellis in 72, Cocks a few months later

Public Key Encryption

RSA Cryptosystem

Key Generation

Choose large primes: p, q

$$N = p \cdot q$$

$$\phi(N) = (p-1)(q-1)$$

Choose:

$$e \; : \; gcd\,(e, \phi(N)) = 1$$

$$d \; : \; ed \; mod \; \phi(N) = 1$$

Output:

$$pk = (e, N)$$

$$sk = d$$

Encryption

$$c = m^e \; mod \; N$$

Decryption

$$m = c^d \; mod \; N$$

“Textbook RSA”

- In practice, we don’t use Textbook RSA
  - Fully deterministic (not semantically secure)
  - <u>Malleable</u>

$$c' = c \cdot x^e \ mod \ N$$

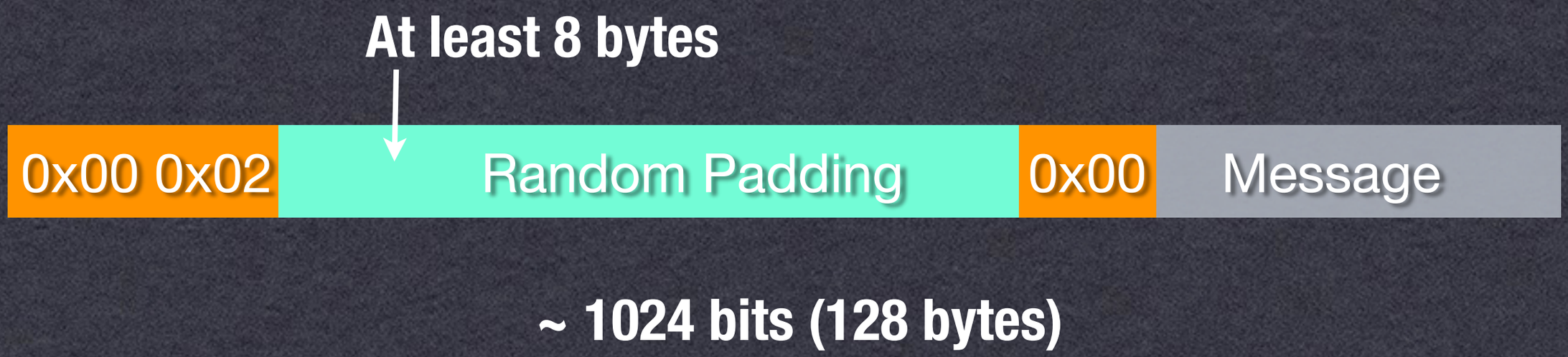
$$c'^d = (m^e \cdot x^e)^d = m \cdot x \ mod \ N$$

  - Might be <u>partially</u> invertible

-Coppersmith’s attack: recover part of plaintext (when m and e are small)

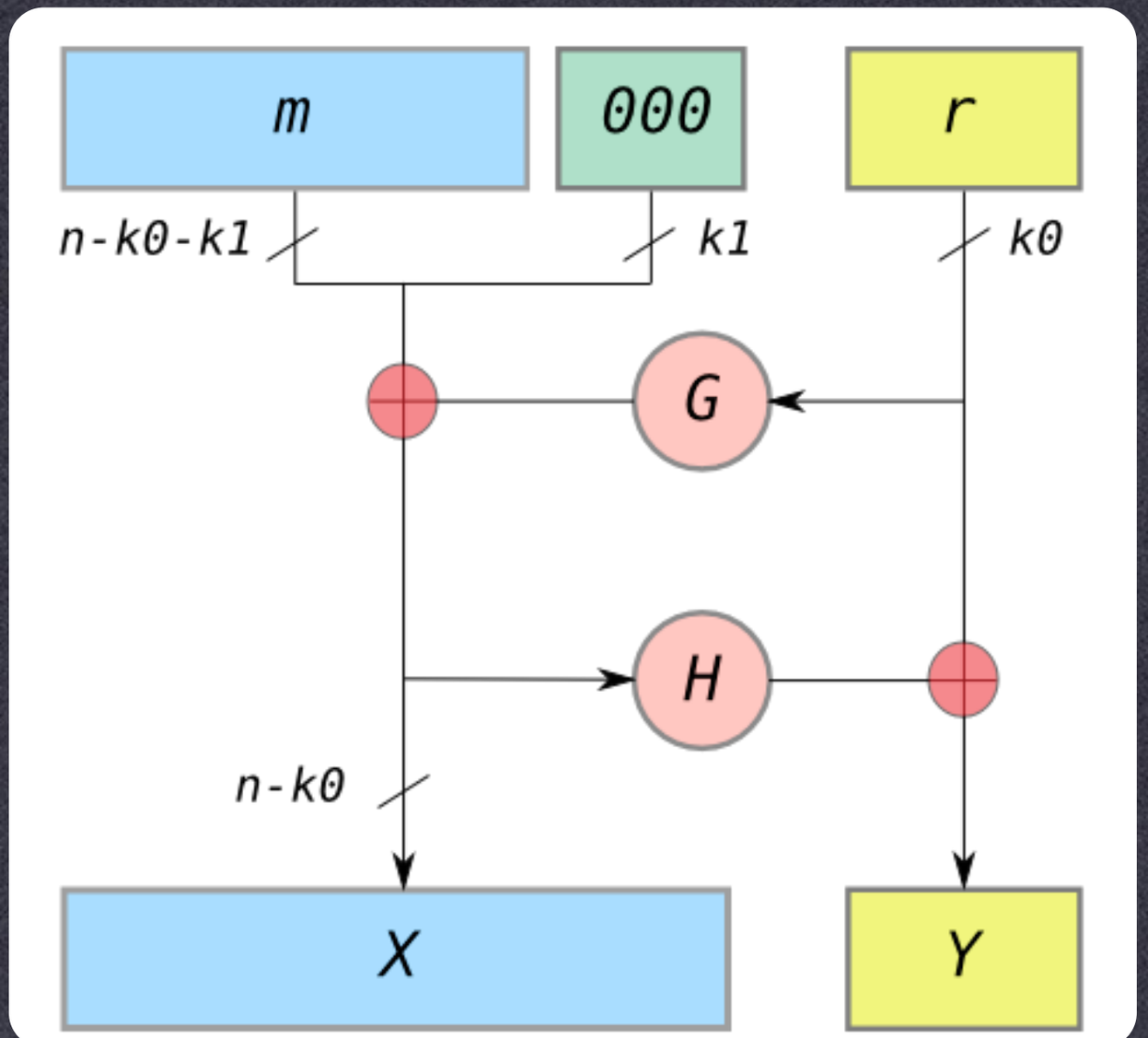
RSA Padding

- Early solution (RSA PKCS #1 v1.5):
  - Add “padding” to the message before encryption
  - Includes randomness
  - Defined structure to mitigate malleability
  - PKCS #1 v1.5 badly broken (Bleichenbacher)



RSA Padding

- **Better solution (RSA-OAEP):**
  - **G and H are hash functions**



Efficiency

	Cycles/Byte
AES (128 bit key)	18
DES (56 bit key)	51
RSA (1024 bit key) Encryption	1,016
RSA (1024 bit key) Decryption	21,719

$m^e \ mod \ N$
$e = 65,537$

$m^d \ mod \ N$

Benchmarks from: http://www.cryptopp.com/benchmarks.html
Microsoft Visual C++, Windows XP, Intel Core 2 1.83Mhz in 32-bit mode

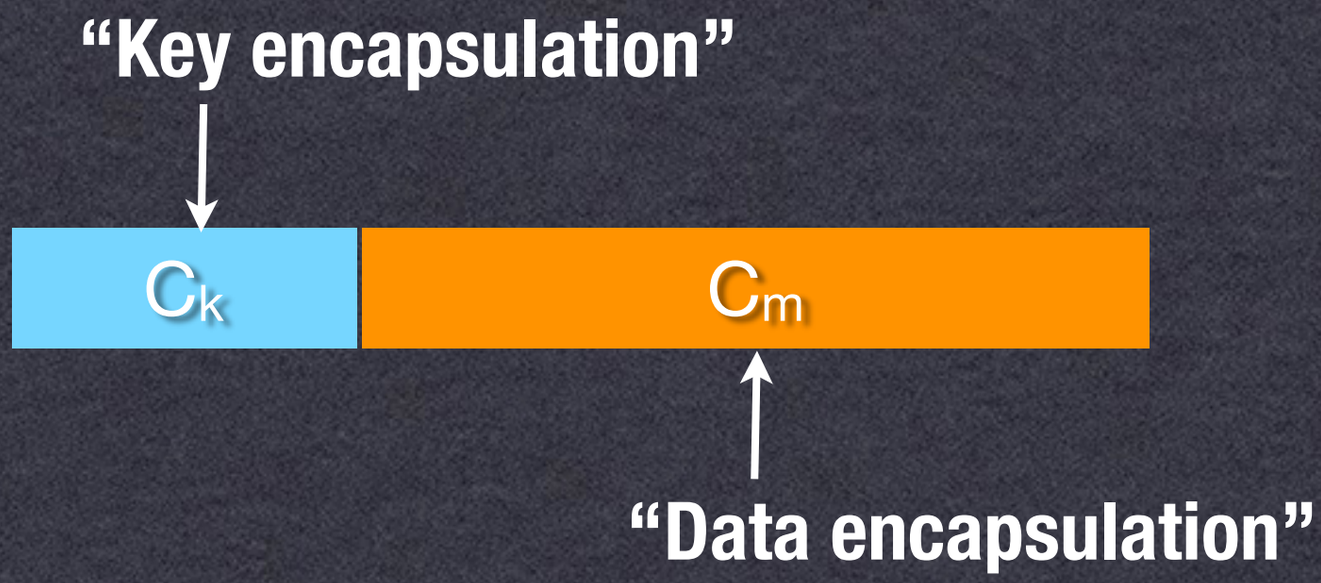
Hybrid Encryption

- **Mixed Approach**
  - **Use PK encryption to encrypt a symmetric key**
  - **Use (fast) symmetric encryption on data**

$$k \xleftarrow{\$} \{0,1\}^k$$

$$C_k \leftarrow RSA.Encrypt_{pk}(k)$$

$$C_m \leftarrow AES.Encrypt_k(message)$$



Key Strength

Level	Protection	Symmetric	Asymmetric	Discrete Logarithm Key	Discrete Logarithm Group	Elliptic Curve	Hash
1	Attacks in "real-time" by individuals *Only acceptable for authentication tag size*	32	-	-	-	-	-
2	Very short-term protection against small organizations *Should not be used for confidentiality in new systems*	64	816	128	816	128	128
3	Short-term protection against medium organizations, medium-term protection against small organizations	72	1008	144	1008	144	144
4	Very short-term protection against agencies, long-term protection against small organizations *Smallest general-purpose level,* *Use of 2-key 3DES restricted to 2^{40} plaintext/ciphertexts,* *protection from 2009 to 2011*	80	1248	160	1248	160	160
5	Legacy standard level *Use of 2-key 3DES restricted to 10^6 plaintext/ciphertexts,* *protection from 2009 to 2018*	96	1776	192	1776	192	192
6	Medium-term protection *Use of 3-key 3DES,* *protection from 2009 to 2028*	112	2432	224	2432	224	224
7	Long-term protection *Generic application-independent recommendation,* *protection from 2009 to 2038*	128	3248	256	3248	256	256
8	"Foreseeable future" *Good protection against quantum computers*	256	15424	512	15424	512	512

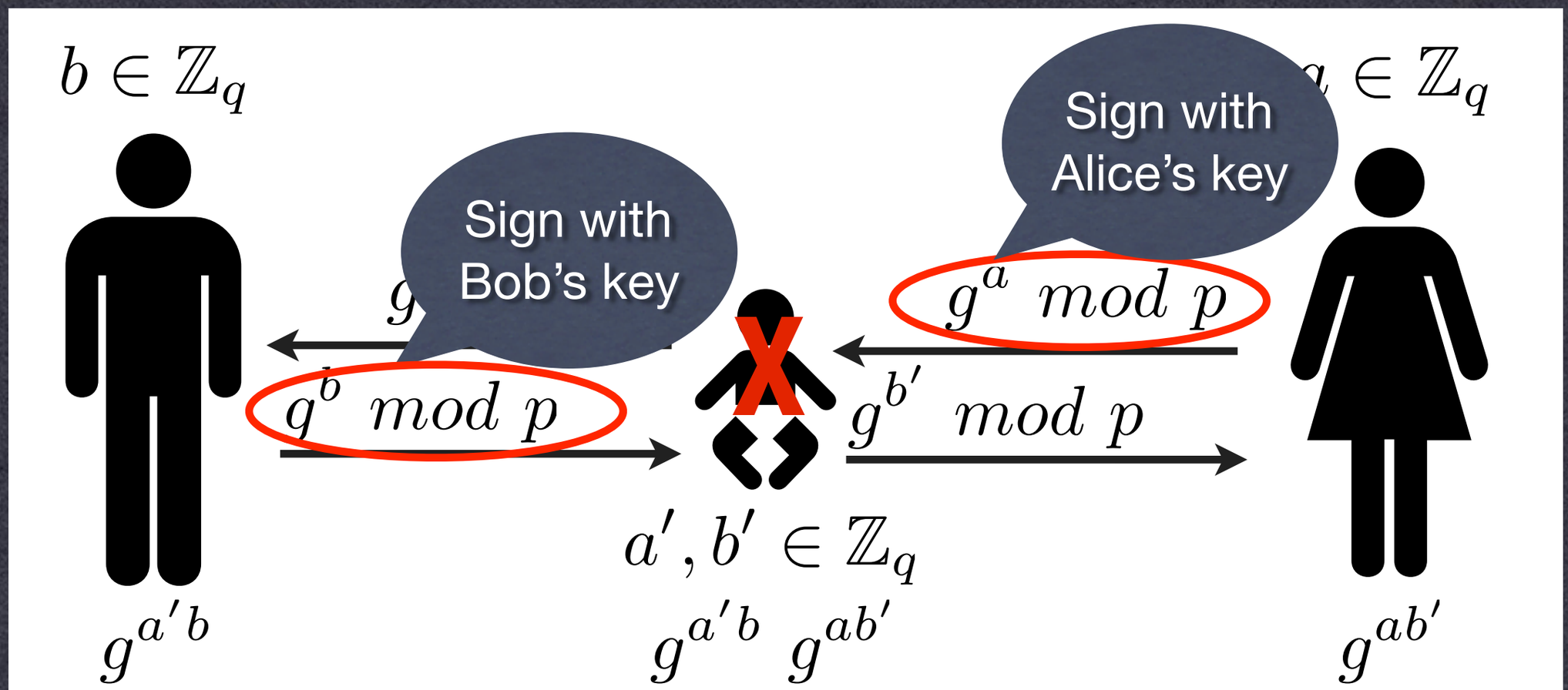
Source: www.keystrength.com (BlueKrypt). Based on ECrypt recommendations.

Digital Signatures

- Similar to MACs, with public keys
  - Secret key used to sign data
  - Public key can verify signature
  - Advantages over MACs?

Preventing MitM

- **Assume an <u>active</u> adversary:**



PKI & Certificates

- How do I know to trust your public key?
  - Put it into a file with some other info, and get someone else to sign it!

Next Time

- Protocols & Implementation
- Reading!
- A2 coming up this week