

Lecture 6 - Cryptography

CSE497b - Spring 2007

Introduction Computer and Network Security

Professor Jaeger

www.cse.psu.edu/~tjaeger/cse497b-s07

Question

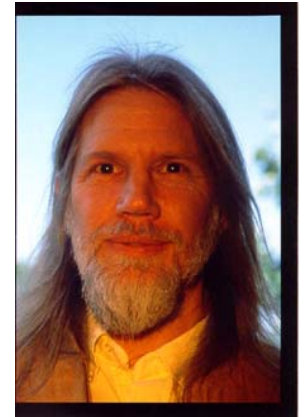
Setup: Assume you and I don't know anything about each other, but we want to communicate securely. We want to establish a key that we can encrypt communication with each other.

Q: Is this possible?



Diffie-Hellman Key Agreement

- The DH paper really started the modern age of cryptography, and indirectly the security community
 - Negotiate a secret over an insecure media
 - E.g., “in the clear” (seems impossible)
 - Idea: participants exchange intractable puzzles that can be solved easily with additional information.



- Mathematics are very deep
 - Working in multiplicative group G
 - Use the hardness of computing discrete logarithms in finite field to make secure
 - Things like RSA are variants that exploit similar properties

Diffie-Hellman Protocol

- For two participants p^1 and p^2
- Setup: We pick a prime number p and a base g ($<p$)
 - This information is public
 - E.g., $p=13$, $g=4$
- Step 1: Each principal picks a private value x ($<p-1$)
- Step 2: Each principal generates and communicates a new value

$$y = g^x \text{ mod } p$$

- Step 3: Each principal generates the secret shared key z

$$z = y^x \text{ mod } p$$

Where y is the value received from the other party.

A protocol run ...

$$p=17, g=6$$

Step 1)

Alice picks $x=4$

Bob picks $x=5$

Step 2)

$$\text{Alice's } y = 6^4 \bmod 17 = 1296 \bmod 17 = 4$$

$$\text{Bob's } y = 6^5 \bmod 17 = 7776 \bmod 17 = 7$$

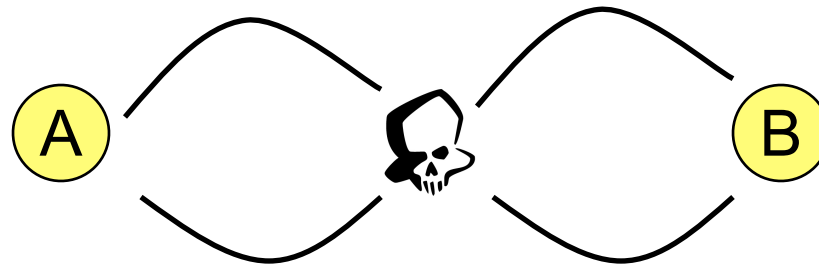
Step 3)

$$\text{Alice's } z = 7^4 \bmod 17 = 2401 \bmod 17 = 4$$

$$\text{Bob's } z = 4^5 \bmod 17 = 1024 \bmod 17 = 4$$

Attacks on Diffie-Hellman

- This is key exchange, not authentication.
 - You really don't know anything about who you have exchanged keys with
 - The man in the middle ...



- Alice and Bob think they are talking **directly** to each other, but Mallory is actually performing two separate exchanges
- You need to have an authenticated DH exchange
 - The parties sign the exchanges (more or less)
 - See Schneier for a intuitive description

Public Key Cryptography

- Public Key cryptography
 - Each key pair consists of a public and private component:
 k^+ (public key), k^- (private key)

$$D(k^+, E(k^-, p)) = p$$

$$D(k^-, E(k^+, p)) = p$$
- Public keys are distributed (typically) through public key certificates
 - Anyone can communicate secretly with you if they have your certificate
 - E.g., SSL-based web commerce

RSA (Rivest, Shamir, Adelman)

- A dominant public key algorithm
 - The algorithm itself is conceptually simple
 - Why it is secure is very deep (number theory)
 - Use properties of exponentiation modulo a product of large primes

"A method for obtaining Digital Signatures and Public Key Cryptosystems", Communications of the ACM, Feb., 1978 21(2) pages 120-126.

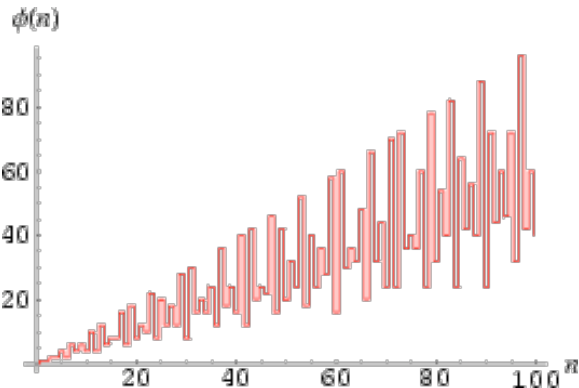


RSA Key Generation

- Pick two large primes p and q
- Calculate $n = pq$
- Pick e such that it is relatively prime to $\phi(n) = (q-1)(p-1)$
 - “Euler’s Totient Function”
- $d \sim e^{-1} \pmod{\phi(n)}$

or

$$de \pmod{\phi(n)} = 1$$



1. $p=3, q=11$
2. $n = 3*11 = 33$
3. $\phi(n) = (2*10) = 20$
4. $e = 7 \mid \text{GCD}(20,7) = 1$
“Euclid’s Algorithm”
5. $d = 7^{-1} \pmod{20}$
 $d = 7 \pmod{20} = 1$
 $d = 3$

RSA Encryption/Decryption

- Public key k^+ is $\{e,n\}$ and private key k^- is $\{d,n\}$
- Encryption and Decryption
 - $E(k^+,P) : \text{ciphertext} = \text{plaintext}^e \bmod n$
 - $D(k^-,C) : \text{plaintext} = \text{ciphertext}^d \bmod n$
- Example
 - Public key (7,33), Private Key (3,33)
 - Data “4” (encoding of actual data)
 - $E(\{7,33\},4) = 4^7 \bmod 33 = 16384 \bmod 33 = 16$
 - $D(\{3,33\},16) = 16^3 \bmod 33 = 4096 \bmod 33 = 4$

Encryption using private key ...

- Encryption and Decryption

$$E(k^-, P) : \text{ciphertext} = \text{plaintext}^d \bmod n$$

$$D(k^+, C) : \text{plaintext} = \text{ciphertext}^e \bmod n$$

- E.g.,

- $E(\{3, 33\}, 4) = 4^3 \bmod 33 = 64 \bmod 33 = 31$

- $D(\{7, 33\}, 19) = 31^7 \bmod 33 = 27,512,614,111 \bmod 33 = 4$

- Q: Why encrypt with private key?

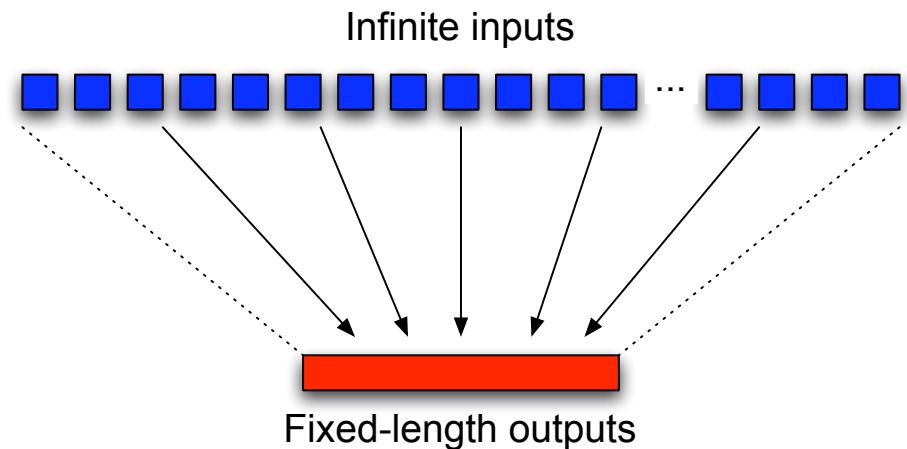
The symmetric/asymmetric key tradeoff

- Symmetric (shared) key systems
 - Efficient (Many MB/sec throughput)
 - Difficult key management
 - Kerberos
 - Key agreement protocols
- Asymmetric (public) key systems
 - Slow algorithms (so far ...)
 - Easy key management
 - PKI - public key infrastructures
 - Webs of trust (PGP)



Hash Algorithms (aka crypto checksums)

- Hash algorithm $h()$
 - In general algorithmic use, generates succinct representation of some data, fixed output size
 - Used for binning items in collections
 - A “funneling algorithm”



- Pigeonhole Principle
 - If you have n bins, and $n+1$ items, at least one bin will contain more than one item
 - Implication: there will be *collisions* in any hash algorithm
 - i.e., $h(x) == h(y)$, for some infinite number of x and y

Hash Algorithms (aka crypto checksums)

- Hash algorithm
 - Compression of data into a hash value
 - E.g., $h(d) = \text{parity}(d)$
 - Such algorithms are generally useful in programs
- ... as used in cryptosystems
 - *One-way* - (computationally) hard to *invert* $h()$, i.e., compute $h^{-1}(y)$, where $y=h(d)$
 - *Collision resistant* hard to find two data x_1 and x_2 such that $h(x_1) == h(x_2)$
- Q: What can you do with these constructs?



Birthday Attack

- A birthday attack is a name used to refer to a class of brute-force attacks.
 - birthday paradox : the probability that two or more people in a group of 23 share the same birthday is >than 50%
- General formulation
 - function $f()$ whose output is uniformly distributed
 - On repeated random inputs $n = \{ n_1, n_2, , \dots, n_k \}$
 - $\Pr(n_i = n_j) = 1.2k^{1/2}$, for some $1 \leq i, j \leq k$, $1 \leq j < k$, $i \neq j$
 - E.g., $1.2(365^{1/2}) \approx 23$
- Q: Why is resilience to birthday attacks important?



Basic truths of cryptography ...

- Cryptography is not frequently the source of security problems
 - Algorithms are well known and widely studied
 - Use of crypto commonly is ... (e.g., WEP)
 - Vetted through crypto community
 - Avoid any “proprietary” encryption
 - Claims of “new technology” or “perfect security” are almost assuredly **snake oil**

Important principles

- Don't design your own crypto algorithm
 - Use standards whenever possible
- Make sure you understand parameter choices
- Make sure you understand algorithm interactions
 - E.g. the order of encryption and authentication
 - Turns out that authenticate then encrypt is risky
- Be open with your design
 - Solicit feedback
 - Use open algorithms and protocols
 - Open code? (jury is still out)

Common issues that lead to pitfalls

- Generating randomness
- Storage of secret keys
- Virtual memory (pages secrets onto disk)
- Protocol interactions
- Poor user interface
- Poor choice of key length, prime length, using parameters from one algorithm in another

Review: secret vs. public key crypto.

- Secret key cryptography
 - Symmetric keys, where A single key (k) is used is used for E and D

$$D(k, E(k, p)) = p$$

- All (intended) receivers have access to key
- Note: Management of keys determines who has access to encrypted data
 - E.g., password encrypted email
- Also known as symmetric key cryptography

- Public key cryptography
 - Each key pair consists of a public and private component: k^+ (public key), k^- (private key)

$$D(k^-, E(k^+, p)) = p$$

$$D(k^+, E(k, -p)) = p$$

- Public keys are distributed (typically) through public key certificates
 - Anyone can communicate secretly with you if they have your certificate
 - E.g., SSL-base web commerce

A really good book on the topic

- The Code Book, Simon Singh, Anchor Books, 1999.

