Chris Graves Introduction to Cryptography

Classic Problem: Alice wants to send Bob a secret message through an insecure channel, but doesn't want anyone to read the message.

General Solution: **Cryptography** Alice should encrypt the message in a way that only Bob can decrypt it and looks like garbage to everyone else

Definitions

$$C = e_k(m) \qquad m = d_k(C)$$

- C: ciphertext
- e_k: encryption function w/ key
- m: message
- d_k: decryption function w/ key

- Assume that Alice and Bob can meet before hand and share a secret key together, this is known as **private key cryptography**

Physical Analogy: Alice and Bob meet together and produce locks with multiple keys that they hold on to, so Alice can lock her message and Bob can open it with his copy of the key.

Private Key Code 1: Substitution Cipher (Roman Empire)

Alice and Bob agree to a 1–1 function F that maps the 'alphabet' to a random permutation of the alphabet

A	B	C	D	E	...
↓	↓	↓	↓	↓	
F	W	Y	A	B	

Pro	Con
Easy to do	Can be cracked by hand, by using letter frequency

Private Key 2: One-Time Pad (1920's)

Alice and Bob share a secret, random binary key K that is at least as long as the message. Alice would convert her message to a binary string m using ASCII. Alice then encrypts m by xoring with K

XOR

$0 \oplus 0 = 0$
$0 \oplus 1 = 1$
$1 \oplus 0 = 1$
$1 \oplus 1 = 0$

$$C = e_K(m) = m \oplus K$$

$$\begin{array}{rl} m: & 101011 \\ \oplus\ K: & 011010 \\ \hline C: & 110001 \end{array}$$

To decrypt, Bob takes C and xor's it with K

$$m = d_K(C) = C \oplus K$$

$$\begin{array}{rl} C: & 110001 \\ K: & 011010 \\ \hline m: & 101011 \end{array} \checkmark$$

works b/c $m = C \oplus K = m \oplus K \oplus K = m \oplus 0 = m$ ✓

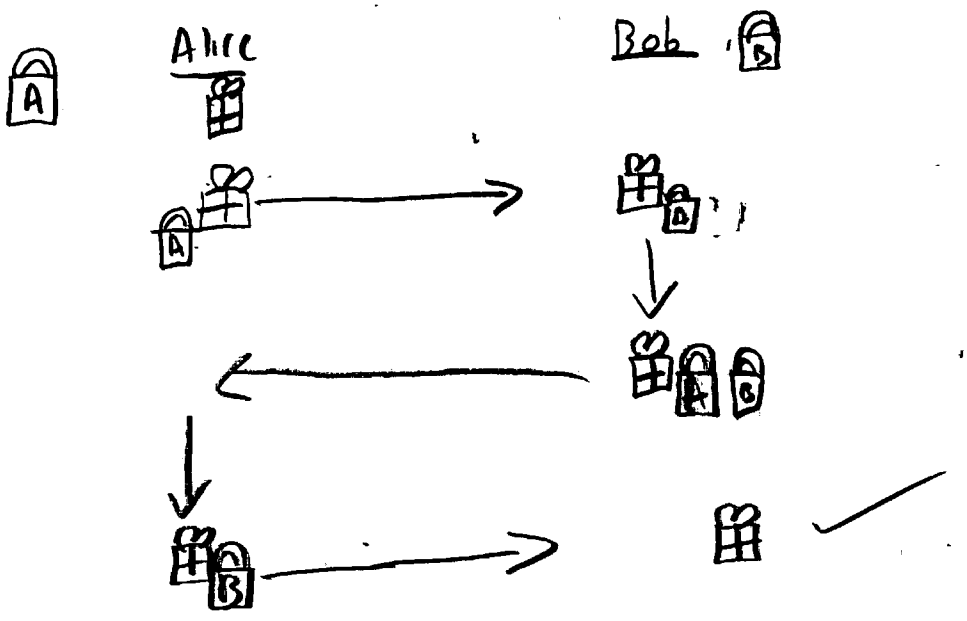
Unless you know the keys the code is unbreakable because a given cipher text is equally likely to be mapped to any message.

Pro	Con
Unbreakable	Need to meet before hand to share secret key

→ Public-Key Cryptography

Don't assume Alice and Bob have any shared secret keys to begin with.

Physical Example



* Point is that all transfered information is "locked" by at least one lock

Diffie Hellman Key Exchange (1976)

- Protocol For two parties to generate a shared secret key (which can be used for one-time pa

Alice

starts with: large Prime p
- base g
- secret integer a

1) Calculates: $A = g^a \bmod p$ (modular exponentiation) by squaring

② p, g, A →

Bob

③ calculates $B = g^b \bmod p$

⑤ $K = A^b \bmod p$

$= g^{ab} \bmod p$ ✓

← ④ B

6) $K = B^a \bmod p$

$= g^{ba} \bmod p$ ✓

Since discrete logarithms for $(\mathbb{Z}_p)^\times$ are difficult, outsiders cannot derive K, given p, g, A, B.

Pro	Con
Don't need shared key before hand	Have to do this protocol Before hand, before actually encrypting anything

Physical Example

Bob just mails his lock to Alice, Alice locks her message, sends it to Bob and Bob opens it with his key

RSA

message + public key → ciphertext

ciphertext + private key → message

* Can't get private key from public key

① Bob picks 2 large secret primes p & q

② Bob computes $N = p \cdot q$ and $\phi(N) = (p-1)(q-1)$ → totient

③ Bob chooses encryption exponent e, which satisfies $\gcd(e, \phi) = 1$

④ Bob has a public key (N, e)

⑤ Bob calculates decryption exponent d, s.t. $e \cdot d \equiv 1 \bmod \phi$ * Extended Euclidean Algorithm

⑥ Bob has a private key (d, p, q)

hm.

Now Alice can take Bob's public key (N, e) and encrypt her message

$c = m^e \bmod N$ * modular exponentiation

Bob can decrypt the cipher text with

$m = c^d \bmod N$

Proof $m^{ed} \equiv m \pmod N$

$e \cdot d \equiv 1 \pmod \phi \rightarrow ed = 1 + K\phi$ → prove $m^{ed} \equiv m \pmod p$ and $m^{ed} \equiv m \pmod q$

Two cases: Prove $m^{ed} \equiv m \pmod p$

1) $\gcd(m, p) = 1$

$m^{p-1} \equiv 1 \pmod p$ *Fermat's Little Theorem

$m \cdot m^{(p-1)\cdot k(q-1)} \equiv m \pmod p$

$m^{k\phi+1} = m^{ed} \equiv m \pmod p$ ✓

2) $\gcd(m, p) = p$

$m \equiv 0 \pmod p$

$m^{ed} = 0 \equiv m \pmod p$

Can do same with q to show $m^{ed} \equiv m \pmod q$

go to other side first →

Modular Exponentiation by squaring, binary method

Trying to calculate $b^e \pmod m$

$e = \sum_{i=0}^{n-1} a_i 2^i$ where $a_i = 0$ or 1 $\quad n = \lceil \log e \rceil$

$b^e = b^{\sum_{i=0}^{n-1} a_i 2^i} = \prod_{i=0}^{n-1} (b^{2^i})^{a_i}$

$b^e \pmod m \equiv \prod_{i=0}^{n-1} (b^{2^i})^{a_i} \pmod m$

Example

$b = 42$

$c = 42^{17} \bmod 55$

$17 = 1\cdot1 + 0\cdot2 + 0\cdot4 + 0\cdot8 + 1\cdot16$

$a_0\ a_1\ a_2\ a_3\ a_4$

$b^{2^i} \equiv (b^{2^{i-1}})^2 \pmod m$

$37 = 1\cdot1 + 0\cdot2 + \dots + 1\cdot32$

$a_0 \qquad\qquad a_5$

$37^{37} \bmod 55$

i	$b^{2^i} \pmod m$	a_i	running total = 1
0	$42^2 \pmod m = 42$	1	$1 \cdot 42 \bmod 55 = 42$
1	$42 \cdot 42 \pmod{55} = 4$	0	$42 \cdot 1 = 42$
2	$4 \cdot 4 \pmod{55} = 16$	0	$42 \cdot 1 = 42$
3	$16 \cdot 16 \pmod{55} = 36$	0	$42 \cdot 1 = 42$
4	$36 \cdot 36 \pmod{55} = 31$	1	$(42 \cdot 31) \bmod 55 = 37$

$37 = c$

	37	1	37
0	37	1	
1		6	
2		0	
3		0	
4		0	
5	16	1	$37 \cdot 16 \bmod 55 = 42$ ✓

Extended Euclidean Algorithm

Like the Euclidean Algorithm, finds the GCD of integers a, b but also finds x, y s.t. $ax + by = gcd(a, b)$

In RSA:

we want d s.t. $e \cdot d = 1 \bmod \phi$ and we know e, ϕ, and that $gcd(e, \phi) = 1$

if we find x, y s.t.

$e \cdot x + \phi \cdot y = gcd(e, \phi) = 1$

$e \cdot x = 1 \bmod \phi$

$\therefore (x = d) \bmod \phi$ ✓

$r_i = a x_i + b y_i$

$r_i = r_{i-2} - q_i r_{i-1}$ where $q_i = \left\lfloor \frac{r_{i-2}}{r_{i-1}} \right\rfloor$

$r_1 = a = a \cdot 1 + b \cdot 0$

$r_2 = b = a \cdot 0 + b \cdot 1$

GCD is last r_i before it equals 0

Suppose we have

$p = 5$, $q = 11$, $t = 55$

$\phi = 4 \cdot 10 = 40$

$e = 17$

Run EEA on 17 and 40

$40 = 40 \cdot 1 + 17 \cdot 0$

$17 = 40 \cdot 0 + 17 \cdot 1$

$6 = 40 \cdot 1 + 17 \cdot -2$

$5 = 40 \cdot -2 + 17 \cdot 5$

$1 = 40 \cdot 3 + 17 \cdot -7$

0

$-7 \bmod 40 = 33 = d$

$e \cdot d = 1 \bmod \phi$

$17 \cdot 33 = 561 \bmod 40$

$= 1 \bmod 40$ ✓