The Miracle of Public-Key Cryptography

Edward Schaefer
Department of Mathematics and Computer Science, Santa Clara University

The Spartan scytale cipher (5th century BC) is an example of a *transposition* cipher because the symbols in the message are simply rearranged.

Julius Caeasar and his generals used the following cipher. They would shift each letter three to the right. In our alphabet that would be

$$\begin{array}{ccccccccccccc} A & B & C & D & E & F & G & H & I & J & K & L & M \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ D & E & F & G & H & I & J & K & L & M & N & O & P \end{array}$$

$$\begin{array}{ccccccccccccc} N & O & P & Q & R & S & T & U & V & W & X & Y & Z \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ Q & R & S & T & U & V & W & X & Y & Z & A & B & C \end{array}$$

So *plaintext* COMMENCE would become
ciphertext FRPPHQFH. To decrypt the message, you just shift each letter three to the left. So $D \to A$, etc.

We call this a *substitution* cipher since we are replacing some letters by others.

Slightly safer cipher: vary the shift.

On Tuesday, shift 17 to the right to encrypt and 17 to the left to decrypt.

The *encrypting key* is +17 and the
decrypting key is −17.

For Caesar they were +3 and −3.

For the Greek scytale cipher the encrypting and decrypting keys would be the same: the diameter of the scytale.

Germany Army's ADFGVX cipher, WWI.
Early *product cipher*. So alternates substitution and transposition.

There was a fixed table.

	A	D	F	G	V	X
A	K	Z	W	R	1	F
D	9	B	6	C	L	5
F	Q	7	J	P	G	X
G	E	V	Y	3	A	N
V	8	O	D	H	0	2
X	U	4	I	S	T	M

Replace plaintext letter by pair (row, column).

Plaintext PRODUCTCIPHERS becomes

FG AG VD VF XA DG XV DG XF FG VG GA AG XG.

That's the substitution.

PRODUCTCIPHERS →
FG AG VD VF XA DG XV DG XF FG VG GA AG XG.

Transposition part follows and depends on a key.

Let's say the key is DEUTSCH.

Put the cipher so far under the key.
Number the letters alphabetically.

D	*E*	*U*	*T*	*S*	*C*	*H*
2	3	7	6	5	1	4
F	*G*	*A*	*G*	*V*	*D*	*V*
F	*X*	*A*	*D*	*G*	*X*	*V*
D	*G*	*X*	*F*	*F*	*G*	*V*
G	*G*	*A*	*A*	*G*	*X*	*G*

Write the letters in numerical order by columns. Ciphertext: DXGX FFDG GXGG VVVG VGFG GDFA AAXA

In WWII found alternating substitution and transposition is very secure.

ADFGVX is weak since the subst'n and trans'n

each occur once and subst'n fixed, not key controlled.

In WWII, complicated product ciphers on machines such as ENIGMA (German) and PURPLE (Japanese) were used.

This continues. DES (the Data Encryption Standard) for computers introduced in 1975.

The plaintext is turned into 0's and 1's (bits).

ASCII: there is an 8 bit representation for each symbol
A=01000001
B=01000010
⋮
a=01100001
⋮
0=00110000
1=00110001
⋮
?=00111111
 =00100000.

DES uses 56 bit key and sends 64 bit blocks of plaintext to 64 bit blocks of ciphertext.

Note that eight symbols of plaintext would be a $8 \cdot 8 = 64$ bit block.

DES alternates 16 substitutions and with 15 transpositions.

DES turned into triple-DES with a $2 \cdot 56 = 112$ bit key and 64 bit blocks of plaintext and ciphertext.

Triple-DES is being replaced by AES, (the Advanced Encryption Standard) which has a 128 bit key and sends a 128 bit block of plaintext to 128 bit block of ciphertext.

All ciphers mentioned had one key (Greek Baton, ADFGVX, DES, AES) or easy to get decrypting from encrypting key (Caesar).

Problem: Users must agree on the key ahead of time.

This requires being in the same place at the same time (a hassle) or a trusted courier, like a runner or a phone call (unsafe).

Want to use credit card to buy a book at Amazon's website.

Must agree on key for AES to encrypt credit card number before sending over web.

Don't want to travel to their headquarters.

Don't want to call them.

In 1975, Diffie and Hellman of Stanford thought of public key cryptography.

You want to encrypt a message for Amazon with AES.

Amazon creates a private key that only they will know.

From the private key, they create a public key that is posted on their website.

Unlike with classical ciphers, the private key is neither the same, nor easily deduced from their public key.

How is this done?

Modular arithmetic: Time on a clock works mod 12.

If it's 5 hours after 10 then it's 3 o'clock.

So we say that $10 + 5 = 3 (\mathrm{mod}\, 12)$.

That's because if we divide $10 + 5 = 15$ by 12 we get a remainder of 3.

$10 + 5 = 3 (\mathrm{mod}\, 12)$.

Can also multiply mod 12.

Start at noon and take three ten hour flights, what time will we arrive? (Noon$=0$.)

Well $3 \cdot 10 = 30 = 6 (\mathrm{mod}\, 12)$.

So we'll land at 6 o'clock. (Lost info on day, am/pm, don't care).

We could move to planet where clock has 19 hours. (Picked since 19 prime.)

As long as we can multiply, let's look at powers of 2.

Mod 19:

$$\begin{array}{ll}
2^1 = 2 & 2^2 = 4 \\
2^3 = 8 & 2^4 = 16 \\
2^5 = 32 = 13 & 2^6 = 64 = 7 \ (2 \cdot 13 = 26 = 7) \\
2^7 = 2 \cdot 7 = 14 & 2^8 = 28 = 9 \\
2^9 = 18 & 2^{10} = 36 = 17 \\
2^{11} = 34 = 15 & 2^{12} = 30 = 11 \\
2^{13} = 22 = 3 & 2^{14} = 6 \\
2^{15} = 12 & 2^{16} = 24 = 5 \\
2^{17} = 10 & 2^{18} = 20 = 1 \\
2^{19} = 2 & 2^{20} = 4 \text{ etc.}
\end{array}$$

i	1	2	3	4	5	6	7	8	9	10
2^i mod 19	2	4	8	16	13	7	14	9	18	17

i	11	12	13	14	15	16	17	18
2^i mod 19	15	11	3	6	12	5	10	1

Second row somewhat random. Note second row is a permutation of $\{1, 2, \ldots, 18\}$.

Powers of 2 don't give a permutation for all primes.

For example for prime $p = 7$: $2^1 = 2$, $2^2 = 4$, $2^3 = 8 = 1$, $2^4 = 2$, etc. Don't get $3, 5, 6$.

We'll stick with primes p for which powers of 2 mod p give permutation of $\{1, 2, \ldots, p-1\}$.

$p = 101$ is prime and powers of 2 give a permut'n $\{1, 2, \ldots, 100\}$ mod 101.

So $2^x = 3 (\bmod 101)$ has a solution.

So $2^x \div 101$ has a remainder of 3 for some x.

What's x?

Solve $2^x = 5 (\bmod 101)$.

That's called solving the discrete logarithm problem.

$2^x = 3 (\bmod 101)$ has solution $x = 69$.

$2^x = 5 (\bmod 101)$ has solution $x = 24$.

How can you solve faster than brute force $2^1 =$, $2^2 =$, ...

Here you'll solve problem in ≤ 100 steps.

What if $p =$ 491995465550325836178627551170573640
9715816438094174259261975937215638823906752013622
8007328858648340078757185950591410307269516585878
8825074468497581235452760957861227635662502820640

05405187303534776190799515222723?

With current algorithms and current computer speeds, solving takes longer than age of universe.

Diffie & Hellman created a key agreement system. Can't encrypt specific message. Can create a key for AES without meeting/courier.

Reminder: What's $(x^y)^z$?

$(7^2)^3 = (7 \cdot 7) \cdot (7 \cdot 7) \cdot (7 \cdot 7) = 7^6 = 7^{2 \cdot 3}$.

Always works. $(x^y)^z = x^{(yz)}$.

Note $(x^y)^z = x^{yz} = (x^z)^y$.

You and Amazon need to agree on AES key.

Amazon's website specifies the prime $p \approx 10^{200}$ for which powers of 2 give a permutation of $\{1, 2, \ldots, p-1\} \pmod{p}$.

You (really your computer)create a random number Y with $1 < Y < p \approx 10^{200}$.
That's private key. Keep secret.

Amazon creates a random number Y (each transaction) with $1 < Y < p \approx 10^{200}$.
That's private key. Keeps secret.

Fix $p \approx 10^{200}$.

You create secret random Y, Amazon creates secret random A.

You compute $2^Y \pmod{p} = r_Y$ and send to Amazon. r_Y called a public key.

Note this number is $< p$ and so is MUCH smaller than 2^Y.

Amazon computes $2^A \pmod{p} = r_A$ and sends it to you. This number is also $< p$.

Fix $p \approx 10^{200}$.

You create secret random Y and send
$[2^Y \pmod{p}] = r_Y$ to Amazon.

Amazon creates secret random A and sends
$[2^A \pmod{p}] = r_A$ to you.

You take r_A and raise it to Y.

Get $(r_A)^Y = [(2^A)^Y] = 2^{AY} \pmod{p} = r_{AY}$.

Amazon takes r_Y and raises it to A.

Gets $(r_Y)^A = [(2^Y)^A] = 2^{YA} \pmod{p} = r_{AY}$.

So you and Amazon have the same shared key: $2^{AY} \pmod{p} = r_{AY}$.

You both write r_{AY} in binary/base 2 and use first 128 bits as AES key.

Use AES to send messages back and forth.

Seems incredible that hackers see p, 2,
$r_Y = [2^Y \pmod p]$ and $r_A = [2^A \pmod p]$.

Yet they can't figure out $r_{AY} = [2^{AY} \pmod p]$ (shared key).

How could they? Hacker can figure out
$2^A 2^Y = 2^{A+Y} \pmod p$ - useless.

Seems incredible that hackers see p, 2,
$r_Y = [2^Y \pmod p]$ and $r_A = [2^A \pmod p]$. But can't get $r_{AY} = [2^{AY} \pmod p]$.

To get from $r_A = 2^A \pmod p$ to $2^{AY} \pmod p$ need Y.

How can hacker find Y?

Hacker knows knows p, 2 and $r_Y = [2^Y \pmod p]$ with $1 < r_Y < p \approx 10^{200}$.

So hacker can get Y if she can solve
$2^x = r_Y \pmod p$.

But that is the discrete log problem, for which there is no known fast solution.

Note you and Amazon need never meet or send private keys by courier.

This solves the problem of classical cryptography on how to safely agree on a key conventiently.

ElGamal used this idea for system to send messages as well.

If Alice wants to encrypt a message for Bob, he can send her $2^{a_B} \pmod p$.

Alice uses it to encrypt a message.

The only way to decrypt it is to use the number B, which only Bob knows and which can't be deduced from $2^B \pmod p$ because that's the discrete logarithm problem.

ElGamal used this idea for system to sign messages.

Bob can use his B to electronically sign a message.

Alice can use the $2^B \pmod p$ posted on Bob's website to verify that the signature came from him, because only someone with B could have created the signature.