Encrypted Key Exchange

Steven M. Bellovin

smb@cs.columbia.edu

http://www.cs.columbia.edu/~smb

Columbia University

Analyzing Kerberos

- In 1990, Mike Merritt and I were analyzing *Kerberos*
- Kerberos is a cryptographic network authentication system: you type in your password and it lets you set up encrypted sessions to various servers
- We found a number of flaws, some significant. But we also wondered about the role of the password: was it a security risk?

Passwords are Guessable

- We knew from the literature — and experience — that many people pick bad passwords [Morris and Thompson, 1979]
- Experiments show that 10-35% of passwords are guessable, depending on the environment
- The bad guys could always try lots of login attempts, but that's detectable
- Is there another way to exploit password-guessing?

Let’s Back Up

- What are the enemy’s goals?
- What are the enemy’s powers?
- What is a cryptosystem?

What is Encryption?

- Encryption is a mathematical function that takes *plaintext* and a *key* and and produces *ciphertext*
- Think of the old Caesar cipher, $A \rightarrow D$, $B \rightarrow E$, etc.
- The key is 3 — we shift letters down by 3
- Mathematically, let $A = 0$, $B = 1, \ldots.$
- We'll shift by any amount, not just 3

$$E(p, k) = (p + k) \bmod 26$$

Looking at Modern Cryptosystems

- A cryptosystem is a pair of functions, E and D:
- Encryption takes plaintext and a key and produces ciphertext:

$$E : P \times K \rightarrow C$$

- Decryption maps ciphertext and the key to plaintext:

$$D : C \times K \rightarrow P$$

- For most modern cryptosystems, the labels of E and D are arbitrary:

$$\begin{aligned} \forall p, k : D(E(p, k), k) &= p \\ \forall p, k : E(D(p, k), k) &= p \end{aligned}$$

- The output of a good encryption algorithm looks very random.
- Decrypting ciphertext with the wrong key produces random garbage

The Enemy

- The enemy knows everything about how your system works, but doesn't know your passwords
- The enemy has complete control of the network
- You hand your packets to the enemy for delivery
- The enemy can inspect, modify, delete, or repeat any or all packets
- The enemy wants to get your password; with that, not only can all conversations, past and present be read, the enemy can impersonate you

What the Enemy Actually Sees

- At some point, you send or receive a message encrypted using your password as a key:

$$E(M, \text{PW})$$

- Can the enemy attack this?
- Passwords are guessable — let's guess at keys

Guessing at Keys

Suppose you see the message **uryyb jbeyq**, encrypted with a Caesar cipher. What's the key?

Key	*Plaintext*
Key 1	tqxxa iadxp
Key 2	spwwz hzcwo
Key 3	rovvy gybvn
...	
Key 11	jgnnq yqtnf
Key 12	ifmmp xpsme
Key 13	hello world
Key 14	gdkkn vnqkc
Key 15	fcjjm umpjb
...	

You Don't Need to Know the Language!

Suppose the ciphertext is **jwvrwcz tm uwvlm**.

Key	*Plaintext*
. . .	
Key 5	erqmrxu oh prqgh
Key 6	dqplqwt ng oqpfg
Key 7	cpokpvs mf npoef
Key 8	bonjour le monde
Key 9	anmintq kd lnmcd
. . .	

Verifiable Plaintext

- Most real plaintext is easily recognizable
- (Some Unix systems have a command `caesar` that will automatically try to find the key to Caesar ciphers)
- This is called *verifiable plaintext*
- If a message has verifiable plaintext, an attacker can validate guesses at passwords
- We need a way to encrypt messages, using a password as a key, that does not have any verifiable plaintext

Another Detour into Math

- Given b^x, can we find x?
- Sure — that's $\log_b x$
- But what if we have $b^x \bmod p$?
- That's called the *discrete log* problem, and there are no efficient solutions for large enough (about 1024 bits) moduli

Diffie-Hellman Key Exchange

- Suppose that A and B — conventionally, Alice and Bob — want to send encrypted messages to each other. They need a key.
- They agree on a base b and a modulus p
- Alice picks a random number x and sends Bob $b^x \bmod p$
- Similarly, Bob picks a random number y and sends Alice $b^y \bmod p$
- Alice knows x and $b^y \bmod p$, and can calculate $(b^y)^x \bmod p \equiv b^{xy} \bmod p$
- Similarly, Bob can calculate $(b^x)^y \bmod p \equiv b^{xy} \bmod p$
- An eavesdropper only knows $b^x \bmod p$ and $b^y \bmod p$, and can't calculate the shared secret!

What Do These Exponentials Look Like?

- Let's look at the powers of $3 \bmod 11$
- 3, 9, 5, 4, 1, 3, 9, 5, 4, 1
- That misses a lot of choices
- But what of the powers of 2 mod 11?
- 2, 4, 8, 5, 10, 9, 7, 3, 6, 1
- That got them all
- If we pick the right base, its exponentials are all the non-zero numbers less than 11
- More generally, if p and q are prime and $p = 2q + 1$, half of the integers less than p are *generators* of the *group* $\mathbb{Z}_p$

We Have our Pieces

- If we pick b and p properly, Diffie-Hellman exponentials are uniformly distributed in the range $[1, p-1]$
- Let's encrypt the exponential with the password:

$$E(b^x \bmod p, \mathrm{PW})$$

- Suppose the attacker guesses at PW
- An incorrect guess yields a uniformly distributed random number
- A correct guess yields a Diffie-Hellman exponential — and that's uniformly distributed, too!
- There is no verifiable plaintext; the attacker gains no information

The Final Result

- Alice and Bob each pick random numbers, and calculate Diffie-Hellman exponentials
- They encrypt these exponentials with the shared password and exchange them:

$$E(b^x \bmod p, \mathrm{PW}) \quad \rightarrow$$
$$\leftarrow \quad E(b^y \bmod p, \mathrm{PW})$$

- They each know the password, so they can decrypt the exponentials and carry out the Diffie-Hellman calculation
- The result can be used to encrypt the rest of the traffic
- An attacker learns nothing, no matter how guessable the password

What Happened Next

- Mike Merritt and I came up with some more schemes, and published a paper
- All of the variants *except* this one — and it was the original — have been cracked. This one is still believed to be secure.
- This paper found the sub-branch of cryptography known as SPAKA — Strong Password Agreement and Key Agreement protocols
- There are now many more ways to solve this problem; some have been formally proven to be secure

References

- Steven M. Bellovin and Michael Merritt, "Encrypted Key Exchange: Password-Based Protocols Secure Against Dictionary Attacks", *Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy*, May 1992. `http://www.cs.columbia.edu/~smb/papers/neke.pdf`.
- Steven M. Bellovin and Michael Merritt, "Augmented Encrypted Key Exchange", *Proceedings of the First ACM Conference on Computer and Communications Security*, Nov. 1993. `http://www.cs.columbia.edu/~smb/papers/aeke.pdf`
- Steven M. Bellovin, "Probable Plaintext Cryptanalysis of the IP Security Protocols", *Proceedings of the Symposium on Network and Distributed System Security*, Feb 1997. `http://www.cs.columbia.edu/~smb/papers/probtxt.pdf`