CS915/435 Advanced Computer Security - Elementary Cryptography

Key Agreement

Roadmap

- Symmetric cryptography
  - Classical cryptographic
  - Stream cipher
  - Block cipher I, II
  - Hash
  - MAC
- Asymmetric cryptography
  - **Key agreement**
  - Public key encryption
  - Digital signature

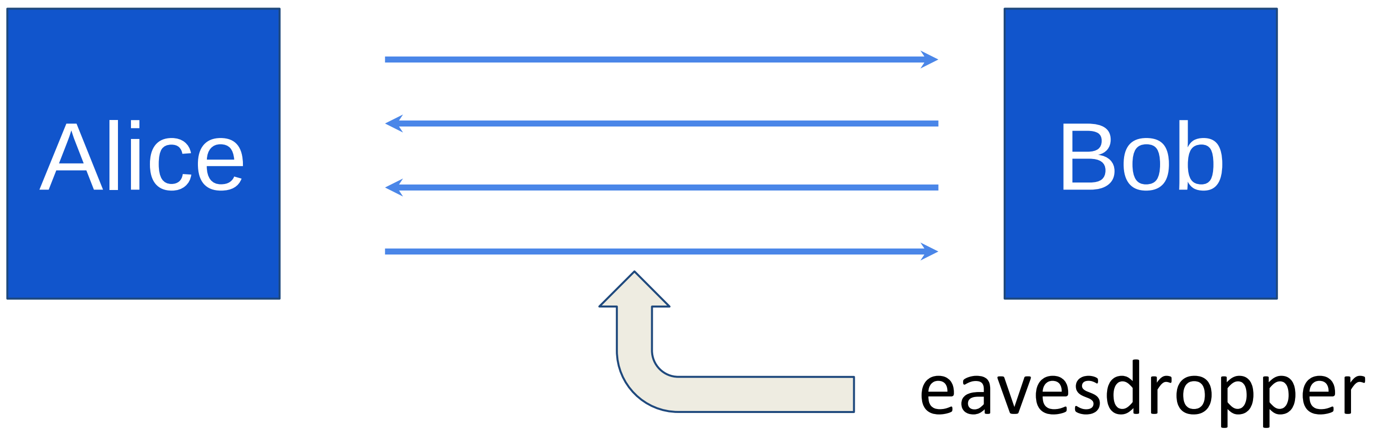
Quote of the day

Good research is done with a shovel, not with tweezers.

— Roger Needham

Goal of key exchange

Alice and Bob want a shared key for secure communication



Is it possible to share a secret key when the eavesdropper can hear everything?

Key exchange in the open air?

- No one had thought it possible
- Until 1974, a Berkeley UG student, Ralph Merkle, proposed the first solution, later known as Merkle puzzles
- He submitted his idea as a project proposal, but his supervisor was not interested, so he went on to Stanford to do a PhD
- He submitted the paper to Communications of ACM, but the paper was harshly rejected

Rejection from CACM

"I am sorry to have to inform you that the paper is not in the main stream of present cryptography thinking and I would not recommend that it be published in the Communications of the ACM."

"Experience shows that it is extremely dangerous to transmit key information in the clear."

Merkle Puzzles (1974)

Main tools: puzzles

- Problems that can be solved with some effort
- E.g. E(k, m) a symmetric cipher with 32-bit k
  - **Puzzle** (P) = E(k, “message”)
  - **Goal**: find k by trying all 2^{32} possibilities

How does it work?

Alice: prepare 2^{32} puzzles

Identifier

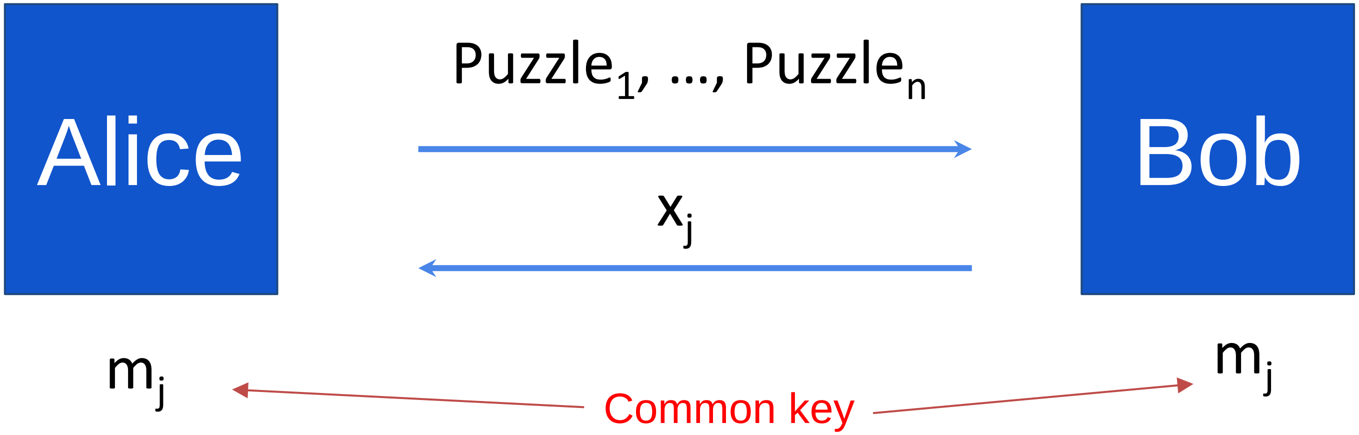
- For $i=1,..,2^{32}$ choose random 32-bit k_i and 128-bit x_i, m_i
- Send all 2^{32} puzzles

key

Bob: choose a random puzzle and solve it. Obtain (x_j, m_j).

- Send x_j to Alice

Alice: lookup puzzle with number x_j. Use m_j as the shared key



Alice’s work: O(n) Prepare n
puzzles

Bob’s work: O(n) Solve one
puzzle

Eavesdropper’s work: O(n^2)

What to learn from an UG idea?

- Merkle's 1974 solution, although inefficient, showed for the first time key exchange in the open air was possible!
- 1976, Diffie and Hellman proposed a more efficient solution, and started a new era in cryptography
- 1978, CACM conceded that rejection was a mistake and published Merkle's paper (keeping the original date in 1975)

> "The human mind treats a new idea the same way the body treats a strange protein; it rejects it."

Peter Medawar

http://www.merkle.com/1974/

Birth of public key cryptography

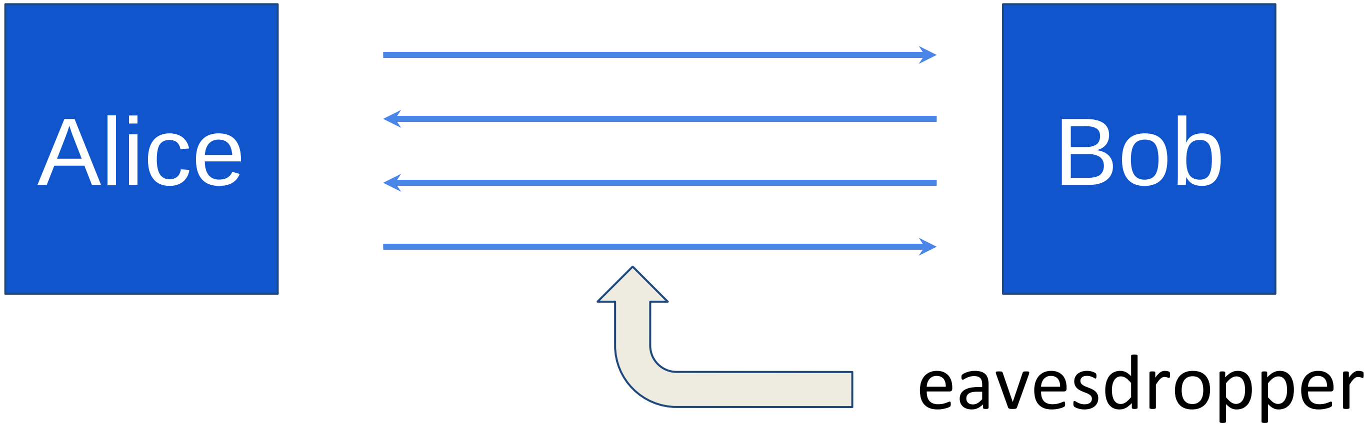
- 1969 - ARPANet born: 4 sites
  - Whitfield Diffie started thinking about secure communication when everyone could read traffic
- 1974 - Whitfield Diffie gave a talk at IBM lab
  - One audience member mentioned that Martin Hellman (Stanford Prof) was working on key distribution
- That night - Diffie started driving 5000 km to Palo Alto
- 1976, Diffie-Hellman key exchange invented

“We stand today on the brink of a revolution in cryptography.”

Diffie and Hellman, “New Directions in Cryptography”, *IEEE Transactions on Information Theory*, Nov 1976.

Key exchange with exponential gap

Merkled showed a solution with quadratic gap



Can this be done with an exponential gap?

Basic discrete logarithm

$g^x \bmod p = y$

- A primitive root modulo p is a number whose powers generate all the nonzero numbers mod p.
- For example, Let p = 7, hence $Z_7^* = \{1, 2, 3, 4, 5, 6\}$

5^1		$= 5 \bmod 7$
5^2	$= 25$	$= 4 \bmod 7$
5^3	$= 4 \times 5$	$= 6 \bmod 7$
5^4	$= 6 \times 5$	$= 2 \bmod 7$
5^5	$= 2 \times 5$	$= 3 \bmod 7$
5^6	$= 3 \times 5$	$= 1 \bmod 7$

Thus, 5 is a primitive root modulo 7

What is the Discrete Logarithm?

Given a value h in $(\mathbf{Z}_p)^*$ with generator g,

find x such that

$$g^{x} = h \ (\mathbf{mod}\ p)$$

Example: $(\mathbf{Z}_{17})^*$, g=3

It's easy to compute $g^8 = 16 \bmod p$

But computing the inverse is difficult

Diffie-Hellman key exchange protocol

- Let p a prime and g a primitive root modulo p

Alice

Select **x** from [1, p-1]

Bob

Select **y** from [1, p-1]

$$A = g^x \quad \longrightarrow$$

$$B = g^y \quad \longleftarrow$$

Compute $K = H(B^x) = H(g^{xy})$

Compute $K' = H(A^y) = H(g^{xy})$

Security

Eve sees: p, g, $A=g^x \pmod p$, $B=g^y \pmod p$

Can she compute $g^{xy} \pmod p$?

More generally: define $DH_g(g^x, g^y) = g^{xy} \pmod p$

How hard is the DH function mod p?

How hard is the DH function?

Suppose prime p is n bits long.

Best known algorithm (GNFS): run time $\exp(O(n^{1/3}))$

Security level	modulus size	Elliptic Curve size
80 bits	1024 bits	160 bits
128 bits	3072 bits	256 bits
256 bits	**15360** bits	512 bits

Slow transition away from (mod p) to elliptic curve

Man-in-the-middle attack

- Let p a prime and g a primitive root modulo p

Alice		Mallory		Bob
Select x from [1, p-1]		Select z from [1,p-1]		Select y from [1, p-1]
	$A = g^x$ →	INTERCEPT	$A' = g^z$ →	
	← $B' = g^z$		← $B = g^y$	
Compute $K = H(g^{xz})$		$K = H(g^{xz})$	$K' = H(g^{yz})$	$K' = H(g^{yz})$

$\boxed{K = H(g^{xz}) = H(B'^x)}$

$\boxed{K' = H(g^{xz}) = H(A'^y)}$

How to prevent the active attack?

- The fundamental limitation with the DH protocol is that it is **unauthenticated**
- Hence, the solution appears simple: let's add authentication!
- Not a trivial problem; over 40 years research
- A very large amount of authenticated key exchange protocols proposed and broken

Two ways to add authenticated

1. Based on **public key certificates**
   - SSL/TLS - Used in https (TLS 1.0, 1.1, 1.2, 1.3)
   - (H)MQV
   - YAK
2. Based on a **password**
   - EKE
   - SPEKE - Used in Blackberry
   - J-PAKE - Used in Google Nest, Thread

Encrypted Key Exchange (1992)

- Each player uses password **s** to encrypt the Diffie-Hellman key exchange process

Alice		Bob
Select x from [1, p-1]	$A = E(s, g^x)$ $\longrightarrow$	Select y from [1, p-1]
	$B = E(s, g^y)$ $\longleftarrow$	
Decrypt B with password		Decrypt A with password
Compute $K = H(B^x) = H(g^{xy})$		Compute $K' = H(A^y) = H(g^{xy})$

“Encrypted Key Exchange: Password-Based Protocols Secure Against Dictionary Attacks", Bellovin and Merritt, IEEE S&P 1992.

Information leakage

Eve captures A, B. She can narrow down the password range.

For s in passwords dictionary

Decrypt A, B

If $D(s, A) \geq p \mid D(s, B) \geq p$

Eliminate s

What went wrong?

The implicit assumption in EKE is that the content in the encryption is random.

But it's not random.

- $A = g^x \bmod p$, the value falls in $[0, p-1]$
- In practice, A is represented as $\{0,1\}^n$, e.g., n=2048
- If the decrypted result falls in $[p, 2^{2048}]$, the candidate password can be ruled out
- The problem is worse if elliptic curve is used