Public Key Encryption Systems

- The encrypter and decrypter have different keys

$$C = E(K_E,P)$$

$$P = D(K_D,C)$$

- Often, works the other way, too

$$C' = E(K_D, P)$$

$$P = D(K_E, C')$$

History of Public Key Cryptography

- Invented by Diffie and Hellman in 1976
- Merkle and Hellman developed Knapsack algorithm in 1978
- Rivest-Shamir-Adelman developed RSA in 1978
  - Most popular public key algorithm
- Many public key cryptography advances secretly developed by British and US government cryptographers earlier

Practical Use of Public Key Cryptography

- Keys are created in pairs
- One key is kept secret by the owner
- The other is made public to the world
- If you want to send an encrypted message to someone, encrypt with his public key
  - Only he has private key to decrypt

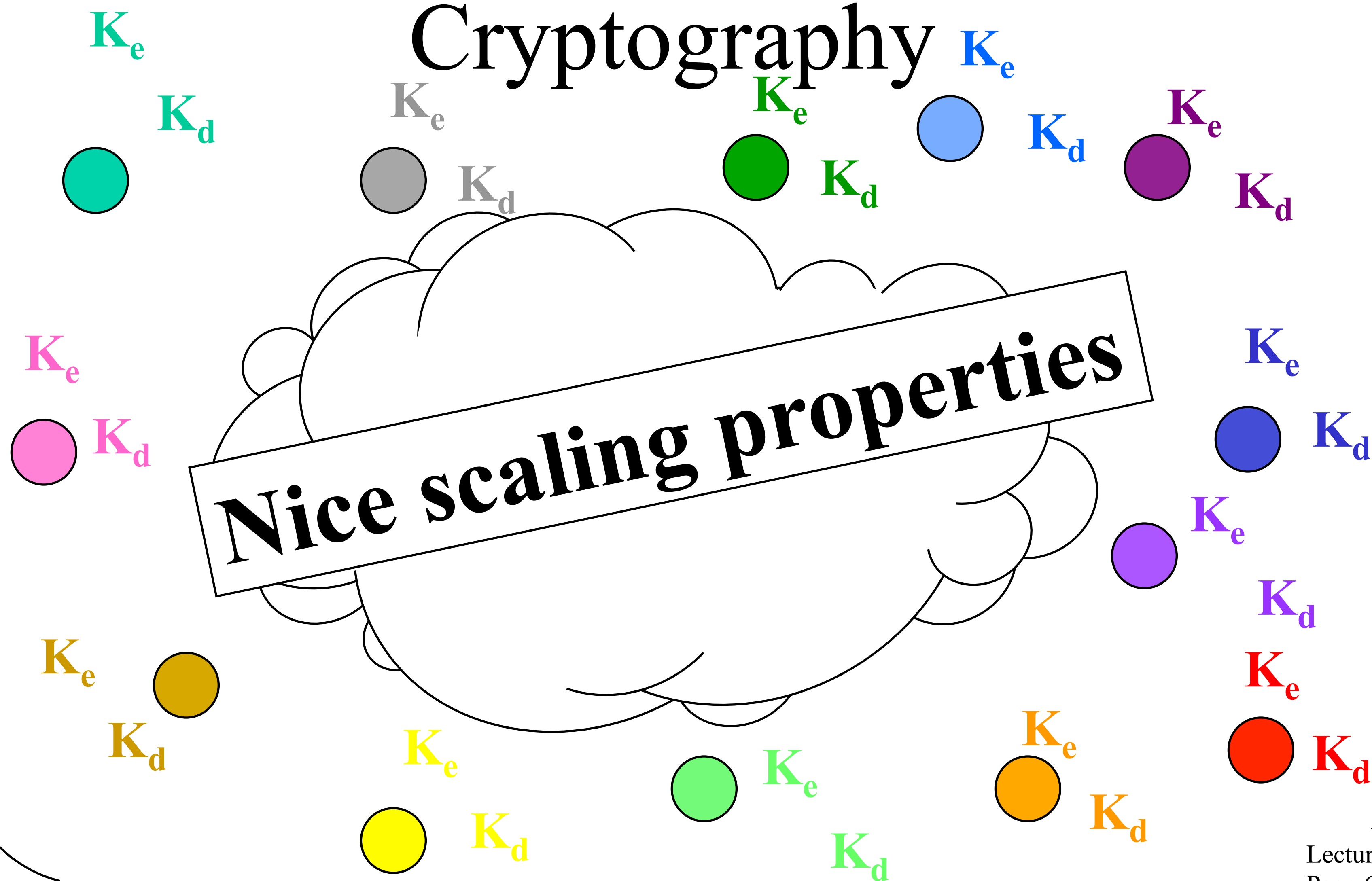
Authentication With Shared Keys

- If only two people know the key, and I didn't create a properly encrypted message -
  - The other guy must have
- But what if he claims he didn't?
- Or what if there are more than two?
- Requires authentication servers

Authentication With Public Keys

- If I want to “sign” a message, encrypt it with my private key
- Only I know private key, so no one else could create that message
- Everyone knows my public key, so everyone can check my claim directly

Scaling of Public Key Cryptography



Key Management Issues

- To communicate via shared key cryptography, key must be distributed
  - In trusted fashion
- To communicate via public key cryptography, need to find out each other's public key
  - "Simply publish public keys"

Issues of Key Publication

- Security of public key cryptography depends on using the right public key
- If I am fooled into using the wrong one, that key's owner reads my message
- Need high assurance that a given key belongs to a particular person
- Which requires a *key distribution infrastructure*

RSA Algorithm

- Most popular public key cryptographic algorithm
- In wide use
- Has withstood much cryptanalysis
- Based on hard problem of factoring large numbers

RSA Keys

- Keys are functions of a pair of 100-200 digit prime numbers
- Relationship between public and private key is complex
- Recovering plaintext without private key (even knowing public key) is supposedly equivalent to factoring product of the prime numbers

Comparison of AES and RSA

- AES is much more complex
- However, AES uses only simple arithmetic, logic, and table lookup
- RSA uses exponentiation to large powers
  - Computationally 1000 times more expensive in hardware, 100 times in software
- RSA key selection also much more expensive

Is RSA Secure?

- Conjectured that security depends on factoring large numbers
  - But never proven
  - Some variants proven equivalent to factoring problem
- Probably the conjecture is correct
- Key size for RSA doesn't have same meaning as DES and AES

Attacks on Factoring RSA Keys

- In 2005, a 663 bit RSA key was successfully factored
- A 768 bit key factored in 2009
- Research on integer factorization suggests keys up to 2048 bits may be insecure
- Insecure key length will only increase
- The longer the key, the more expensive the encryption and decryption

Elliptical Cryptography

- RSA and similar algorithms related to factoring products of large primes
- Other math can be used for PK, instead
  - Properties of elliptical curves, e.g.
- Can give same security as other public key schemes, with much smaller keys
- Widely studied, regarded as safe
  - But the NSA is pushing it . . .
  - Often used for small devices

Combined Use of Symmetric and Asymmetric Cryptography

- Common to use both in a single session
- Asymmetric cryptography essentially used to “bootstrap” symmetric crypto
- Use RSA (or another PK algorithm) to authenticate and establish a *session key*
- Use AES with that session key for the rest of the transmission

Combining Symmetric and Asymmetric Crypto

Alice wants to share the key only with Bob

But there are problems we'll discuss later

Only Bob can decrypt it

Only Alice could have created it

Alice

K_{EA} K_{DA}

K_{EB}

Bob

K_{EB} K_{DB}

K_{EA}

K_S

$C=E(K_S,K_{EB})$

$M=E(C,K_{DA})$

$K_S=D(C,K_{DB})$

$C=D(M,K_{EA})$

Digital Signature Algorithms

- In some cases, secrecy isn't required
- But authentication is
- The data must be guaranteed to be that which was originally sent
- Especially important for data that is long-lived

Desirable Properties of Digital Signatures

- Unforgeable
- Verifiable
- Non-repudiable
- Cheap to compute and verify
- Non-reusable
- No reliance on trusted authority
- Signed document is unchangeable

Encryption and Digital Signatures

- Digital signature methods are based on encryption
- The basic act of having performed encryption can be used as a signature
  - If only I know K, then $C=E(P,K)$ is a signature by me
  - But how to check it?

Signatures With Shared Key Encryption

- Requires a trusted third party
- Signer encrypts document with secret key shared with third party
- Receiver checks validity of signature by consulting with trusted third party
- Third party required so receiver can't forge the signature

For Example,

K_s

When in
the Course
of human
events it
becomes
necessary
for one

Elas7pa
1o’gw0mega
30’sswp.
1f43’-s 4
32.doas3
Dsp5.a#l
^o,a 02

When in
the Course
of human
events it
becomes
necessary
for one

K_s

Signatures With Public Key Cryptography

- Signer encrypts document with his private key
- Receiver checks validity by decrypting with signer's public key
- Only signer has the private key
  - So no trusted third party required
- But receiver must be certain that he has the right public key

For Example,

K_d

When in the Course of human events it becomes necessary for one

Elas7pa 1o'gw0mega 30'sswp. 1f43'-s 4 32.doas3 Dsp5.a#l ^o,a 02

When in the Course of human events it becomes necessary for one

K_e

Alice's public key

Problems With Simple Encryption Approach

- Computationally expensive
  - Especially with public key approach
- Document is encrypted
  - Must be decrypted for use
  - If in regular use, must store encrypted and decrypted versions