Public Key Crypto

Eike Ritter

Computer Security and Networks

This Lecture

- Some more history (not on the exam)
- Diffie Helleman key exchange
- Public Key Encryption
  - Elgamal
  - RSA
  - RSA in practice: keystores, Java & PGP

The Key Problem

- These encryption schemes work well. AES is effectively unbreakable with a “long enough key”.

- The problem is how do you get the key in the first place?

Some History

Before cheap powerful computers, unbreakable encryption was almost impossible.

Governments wanted to read the codes of others.

They could control the export of these machines.

When IBM designed DES they could get it weakened.

- Cipher machines looked like this:

Some History

During 1970-1990 all that changed.

Personal computers could do anything a cipher machine could do.

- Cipher machines looked like this:

Some History

During 1970-1990 all that changed.

Personal computers could do anything a cipher machine could do.

Some History

During 1970-1990 all that changed.

Personal computers could do anything a cipher machine could do.

University academics worked on encryption with the aim of making it available to everyone.

Public Key Encryption

- Public key encryption helps (but doesn't solve) this problem.
- The idea of public key encryption is that you have two keys:
  - one for encryption
  - and another for decryption.
- The encryption key is made public, the decryption key is always secret.

Diffie-Hellman

- Diffie-Hellman is a widely used key agreement protocol.
- It relies on some number theory:
  - a mod b = n where for some “m” : a = m.b + n
- The protocol uses two public parameters
  - generator “g” (often 160 bits long)
  - prime “p” (often 1024 bits long)

See spec doc for e.g. g & p

- http://tools.ietf.org/html/rfc5114
- (section 2)

Diffie-Hellman

- Alice and Bob pick random numbers r_A and r_B and find
  "$t_A = g^{r_A} \bmod p$" and "$t_B = g^{r_B} \bmod p$"

- The protocol just exchanges these numbers:
  1. $A \rightarrow B : t_A$
  2. $B \rightarrow A : t_B$

- "Alice" calculates "$t_B{}^{r_A} \bmod p$" and "Bob" "$t_A{}^{r_B} \bmod p$" this is the key:
  - $K = g^{r_A r_B} \bmod p$

Diffie-Hellman

- An observer cannot work out r_A and r_B from t_A and t_B therefore the attacker cannot calculate the key
- So we have a "**Good Key**" but know nothing about the participants.
- **We did not need to share any keys at the start, therefore this is a very powerful protocol.**
- In practice: use DH to set up a secure channel, then use something else to authenticate the person at the other end.

Example of DH on board.

Elgamal

- Elgamal is Diffie-Hellman turned into public key scheme. It uses a fixed g and p.
- Alice picks r_A as private key
  $t_A = g^{r_A} \bmod p$ is the public key
- Encryption of message M: Bob chooses r_B, sends $(g^{r_B} \bmod p, M * t_A^{r_B} \bmod p)$

RSA

- RSA is a public key system invented by (Rivset, Shamir & Adleman).
- The scheme carefully generates e,d & n
- Public key is (e,n) and the private key is (d,n)

RSA

- Because of how e,d and n are generated:
- To encrypt “m” as the ciphertext “c” do:
  $c = m^e \bmod n$
- To decrypt ciphertext “c” as a message “m” do:
  $m = c^d \bmod n$
- For more details see, the crypto module or Schneier: Applied Cryptography

Using Public Key Crypto

- Public key crypto is much slower than symmetric key.

- So instead of just using public key crypto, systems:
  - make a new symmetric key
  - encrypt that with the public key
  - then encrypt the message with the symmetric key.

Using Public Key Crypto

- Public key crypto is much slower than symmetric key.

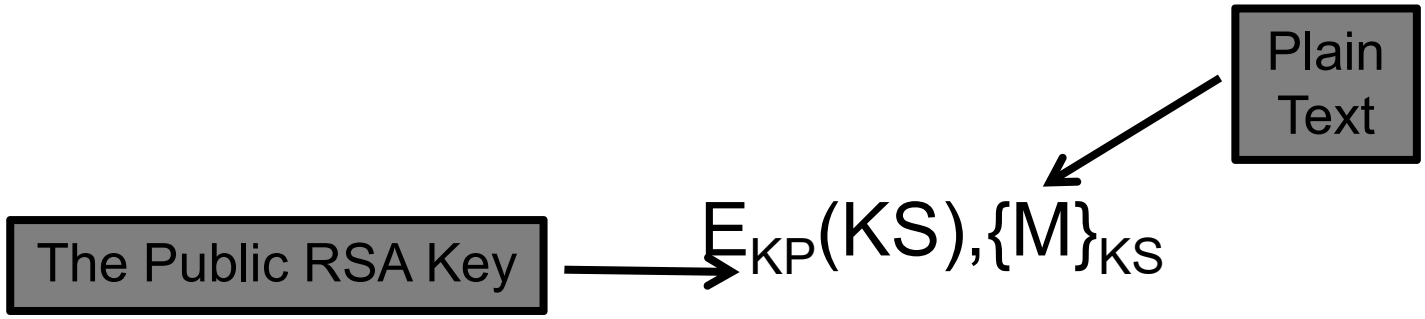
- So instead of just using public key crypto, systems:
  - make a new symmetric key
  - encrypt that with the public key
  - then encrypt the message with the symmetric key.

$$E_{KP}(KS),\{M\}_{KS}$$

Using Public Key Crypto

- Public key crypto is much slower than symmetric key.

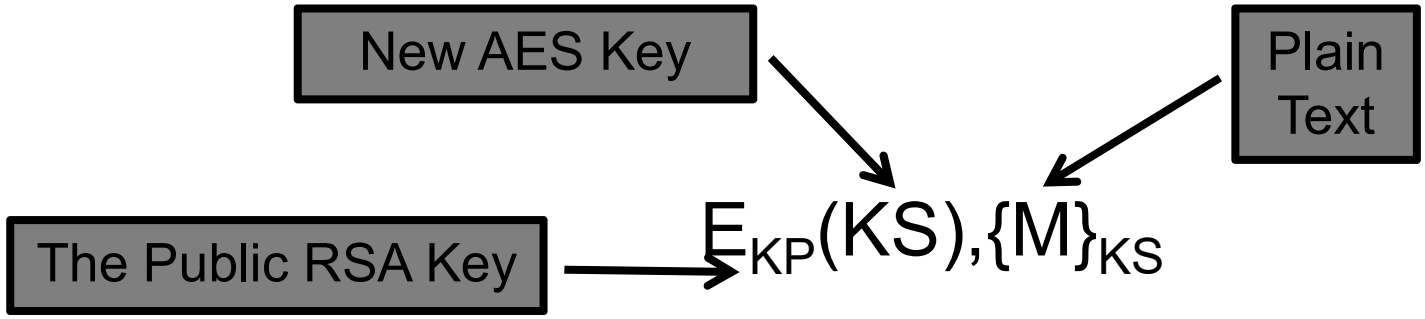
- So instead of just using public key crypto, systems:
  - make a new symmetric key
  - encrypt that with the public key
  - then encrypt the message with the symmetric key.



Using Public Key Crypto

- Public key crypto is much slower than symmetric key.

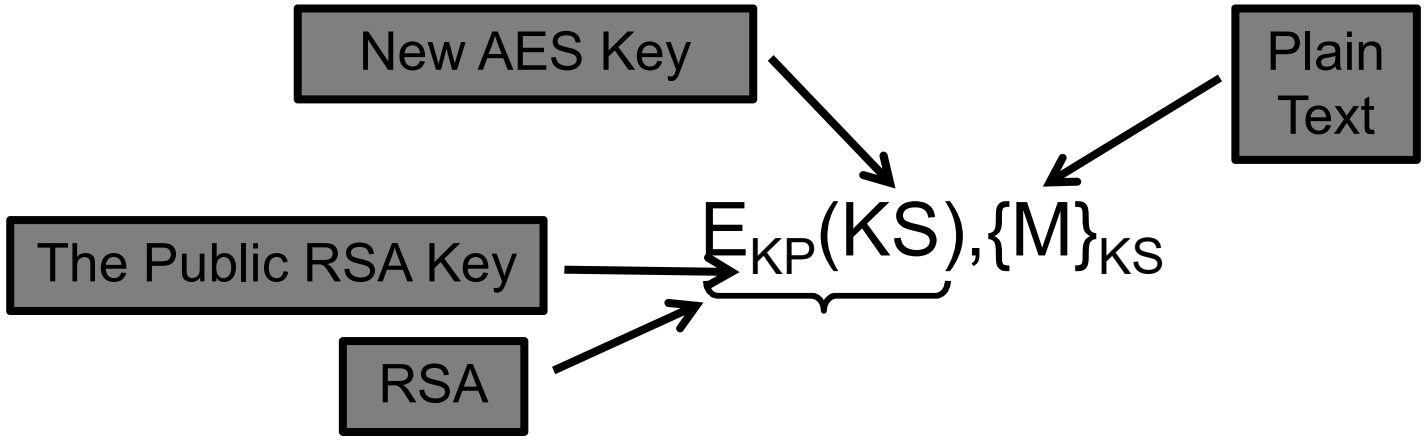
- So instead of just using public key crypto, systems:
  - make a new symmetric key
  - encrypt that with the public key
  - then encrypt the message with the symmetric key.



Using Public Key Crypto

- Public key crypto is much slower than symmetric key.

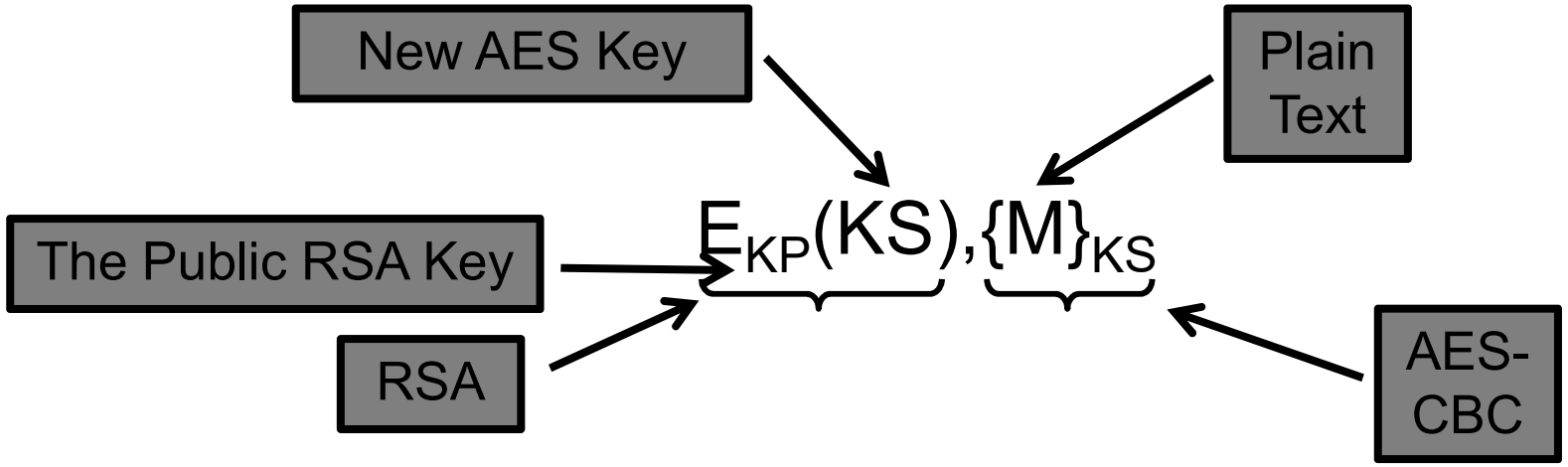
- So instead of just using public key crypto, systems:
  - make a new symmetric key
  - encrypt that with the public key
  - then encrypt the message with the symmetric key.



Using Public Key Crypto

- Public key crypto is much slower than symmetric key.

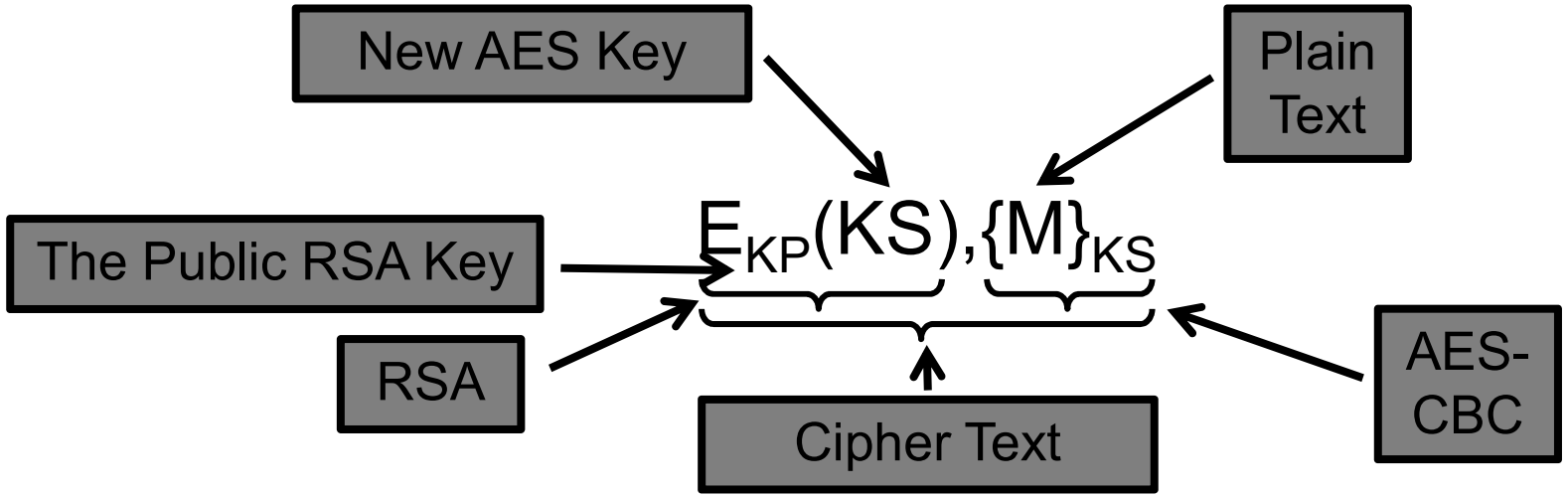
- So instead of just using public key crypto, systems:
  - make a new symmetric key
  - encrypt that with the public key
  - then encrypt the message with the symmetric key.



Using Public Key Crypto

- Public key crypto is much slower than symmetric key.

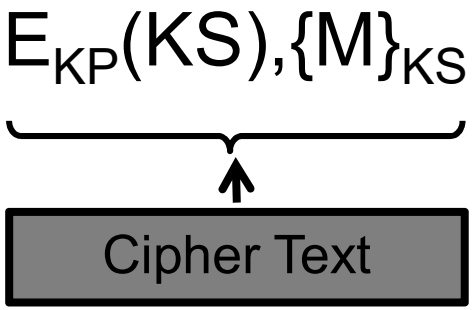
- So instead of just using public key crypto, systems:
  - make a new symmetric key
  - encrypt that with the public key
  - then encrypt the message with the symmetric key.



Using Public Key Crypto

- Public key crypto is much slower than symmetric key.

- So instead of just using public key crypto, systems:
  - make a new symmetric key
  - encrypt that with the public key
  - then encrypt the message with the symmetric key.



Signatures

- Using RSA $E_{pub}(D_{priv}(M)) = M$
- This can be used to sign messages.
- Sign a message with the private key and this can be verified with the public key.
- Any real crypto suite will not use the same key for encryption and signing.
  - as this can be used to trick people into decrypting.

This Lecture

- Some more history (not on the exam)
- Diffie Helleman key exchange
- Public Key Encryption
- RSA
- RSA in practice: keystores, Java & PGP

Saving a Key

- We can read and write the bytes of a key to a file.
  - This is a bad idea.

- We want to
  - protect read access to private keys,
  - and make sure the publics ones are real.

KeyStores and Java keytool

- KeyStore provde password protected storeage for keys.
- Most Java programs use existing keys rather than create keys themselves.
- The keytool command can be used to generate keys outside Java.

The KeyStore Class

- A KeyStore holds password protected private keys and public keys as certicates.
- Make keystores using the keytool e.g.

```
keytool -genkey -keyalg RSA
        -keypass password  -alias mykey
        -storepass storepass
        -keystore myKeyStore
```

Java

Pretty Good Privacy

- In 1991 Phil Zimmermann implemented RSA in an e-mail friendly package.
- He wanted encryption for everyone, especially activists.
- RSA inc. started a licensing dispute.
- The US government started a criminal investigation for arms trafficking!

The Crypto W

Encryption machines like this were classed as “arms”
If key > 40 bits

- Laws in the 1990s were unable to cope with strong encryption from short computer programs.
- Strong crypto available for free on the new Internet panic governments.
- Who was going to control crypto in the age of the Internet?

The Crypto Wars

- 1991: proposed law in the US to ban public key crypto
  - In reaction activists uploaded PGP on to Internet bulleting boards.
- 1993 arms trafficking case against Zimmermann started.
- 1993-1996 Investigation continues,
  - people print RSA algorithm on t-shirts and go through U.S. customs
  - PGP code printed as a book (freedom of speech).
- 1996 Case against Zimmerman dropped
  - But U.S. Attorney: "no change in law, no change in policy”

The Crypto Wars

- 1993-1996: Clipper chip considered in US congress and rejected.
  - Due partly to Matt Blaze's analysis
  - and strongly attack by John Kerry among others.

- 2000 US laws lifted: the Geeks ``won the crypto wars''.

- Freedoms won in the US then filtered through to the rest of the Internet,
  - e.g. French laws until 2004: ECB mode only, max key length 40, must include known plain text.

Crypto Wars: Round 2

- We learnt in 2013 that the NSA had been working to weaken ("back door") crypto.

- Some of the possible backdoors:
  - "Bad" elliptic curve parameters
  - Weak random number generators: e.g. Dual_EC_DRBG

Crypto Wars: Round 2

- Manufacturers added smartphone encryption and end-to-end encryption for apps
- Governments don’t like this: want the equivalent of wiretaps or access to decryption keys for police investigations
- Big problem: weakens crypto or introduces backdoors

PGP

- PGP Demo.

Certificates

- A public key certificate binds a public key to an identity.
- As well as a public key it contains a name, e-mail address, etc.
- It is signed, with the private key, and anyone else that trusts it.

Key Servers

- Key servers store public key certificates
  - E.g. http://pgp.mit.edu, http://keyserver.pgp.com
- Many clients can automatically search a key server for unknown e-mail addresses.
  - But beware, there is no guarantee the key is not a fake.

E-mail demo

https://xkcd.com/364/

Recommended Key Lengths

- See http://www.keylength.com/

This Lecture

- Some more history (not on the exam)
- Diffie-Hellman
- Elgamal
- RSA
- Signing
- Pretty Good Privacy, PGP

Further Reading

See links on the website.

- Coursera Cryptology module
  - www.coursera.org/course/crypto
- Chapter 5 of “Security Engineering”, by Ross Anderson
- Bruce Schneier: Applied Cryptography
- The Code Book, by Simon Singh.

Next Lecture

- Hashes and MAC
- How to security generate short codes from large documents.

Exercise 2

Next Lecture

- Hashes and Macs