CSC 580
Cryptography and Computer Security

Public Key Cryptography - Ideas and RSA
(Related to parts of Chapters 9 and 10)

March 9, 2017

Overview

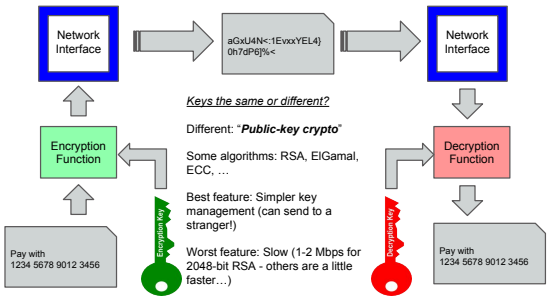
Today:

- HW 7 quiz
- Public Key Algorithms - ideas, math, and RSA

Next:

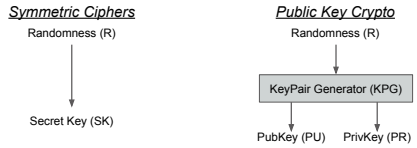
- Spring Break! Have fun!
- If you want to be productive:
  - Work on project phase 3
  - Read Sections 2.4-2.6, 2.8, 10.1, 10.2

Recall Basic Idea



Public Key Crypto

Where do the keys come from?



Mathematical/Computational Properties

- KPG(R) → (PU, PR) is efficiently computatable (polynomial time)
- For all messages M, D(PR, E(PU, M)) = M (decryption works)
- Computing PR from PU is computationally infeasible (we hope!)

Generally: PR has some "additional information" that makes some function of PU easy to compute (which is hard without that info) - this is the "trapdoor secret"

How can this be possible?

To get a sense of how trapdoor secrets help:

Problem: How many numbers $x \in \{1,n-1\}$ have $\gcd(x,N)>1$ for N=32,501,477?
(or: how many have a non-trivial common factor with N?)

How could you figure this out?
How long would it take to compute?
What if N were 600 digits instead of 8 digits?

How can this be possible?

To get a sense of how trapdoor secrets help:

Problem: How many numbers $x \in \{1,n-1\}$ have $\gcd(x,N)>1$ for N=32,501,477?
(or: how many have a non-trivial common factor with N?)

How could you figure this out?
How long would it take to compute?
What if N were 600 digits instead of 8 digits?

What if I told you the prime factorization of N is 5,407 * 6,011?

How can this be possible?

To get a sense of how trapdoor secrets help:

Problem: How many numbers $x \in \{1,n-1\}$ have $\gcd(x,N)>1$ for N=32,501,477?
(or: how many have a non-trivial common factor with N?)

How could you figure this out?
How long would it take to compute?
What if N were 600 digits instead of 8 digits?

What if I told you the prime factorization of N is 5,407 * 6,011?

5,406 multiples of 6,011 share the factor 6,011 with N
6,010 multiples of 5,407 share the factor 5,407 with N
No numbers in common between these two sets (prime numbers!)
So… 5,406+6,010 = 11,416 numbers share a factor with 32,501,477

The factorization of N is a "trapdoor" that allows you to compute some functions of N faster

Using Public Key Crypto in the JCA

Generating a keypair:

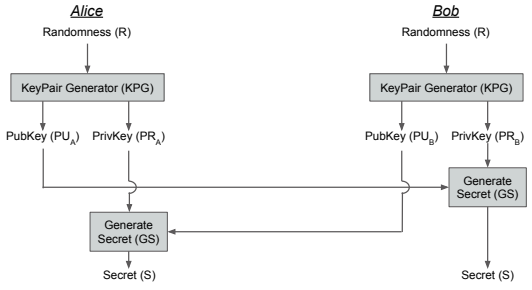
```
public static KeyPair genRSAKey(int bits) {
     KeyPair kp = null;
     try {
          RSAKeyGenParameterSpec kgspec = new RSAKeyGenParameterSpec(bits,
                                                  RSAKeyGenParameterSpec.F4);
          KeyPairGenerator kpg = KeyPairGenerator.getInstance("RSA");
          kpg.initialize(kgspec);
          Kp = kpg.genKeyPair();
     } catch (NoSuchAlgorithmException | InvalidAlgorithmParameterException ex) {
          System.err.println("Oops - basic RSA key generation failed (?)");
     }
     return kp;
}
```

kp.getPublic() gives PublicKey (IS-A Key, so can be used to initialize a Cipher in ENCRYPT_MODE)
kp.getPrivate() gives PrivateKey (IS-A Key, so can be used to initialize a Cipher in DECRYPT_MODE)

Otherwise works just like Cipher with a symmetric cipher algorithm!

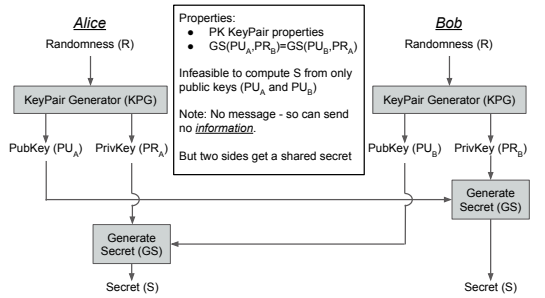
Related Notion - Key Agreement

Original idea - before public key encryption



Related Notion - Key Agreement

Original idea - before public key encryption



Using Key Agreement in the JCA

For algorithm "DH" (Diffie-Hellman)

Generating a keypair requires first generating public parameters:

```
AlgorithmParameterGenerator paramGen =
                        AlgorithmParameterGenerator.getInstance("DH");
paramGen.init(2048);
AlgorithmParameters params = paramGen.generateParameters();
AlgorithmParameterSpec aps = params.getParameterSpec(DHParameterSpec.class);
```

But: Parameter generation can be slow so often done in advance and saved.

➔ Weakness recently found for this, however… be cautious!

Given parameters, Alice and Bob can do key agreement:

```
KeyPairGenerator kpg = KeyPairGenerator.getInstance("DH");
kpg.initialize(aps);
KeyPair aliceKP = kpg.genKeyPair();
KeyAgreement aliceKA = KeyAgreement.getInstance("DH");
aliceKA.init(aliceKP.getPrivate());
aliceKA.doPhase(bobPK, true);
byte[] aliceS = aliceKA.generateSecret();
```

Bob's public key (received)

Key Sizes for Public Key Systems

From NIST publication 800-57a

Issue: PK algorithms based on mathematical relationships, and can be broken with algorithms that are faster than brute force.

We spent time getting a feel for how big symmetric cipher\ keys needed to be

➔ How big do keys in a public key system need to be?

From NIST pub 800-57a:

Table 2: Comparable strengths

Security Strength	Symmetric key algorithms	FFC (e.g., DSA, D-H)	IFC (e.g., RSA)	ECC (e.g., ECDSA)
≤ 80	2TDEA[23]	$L = 1024$ $N = 160$	$k = 1024$	$f = 160$-223
112	3TDEA	$L = 2048$ $N = 224$	$k = 2048$	$f = 224$-255
128	AES-128	$L = 3072$ $N = 256$	$k = 3072$	$f = 256$-383
192	AES-192	$L = 7680$ $N = 384$	$k = 7680$	$f = 384$-511
256	AES-256	$L = 15360$ $N = 512$	$k = 15360$	$f = 512+$

Weakness in long-term fixed DH parameters

From 2015 ACM Conference on Computer and Communication Security:

Imperfect Forward Secrecy: How Diffie-Hellman Fails in Practice

David Adrian[¶] Karthikeyan Bhargavan[*] Zakir Durumeric[¶] Pierrick Gaudry[†] Matthew Green[§]
J. Alex Halderman[¶] Nadia Heninger[‡] Drew Springall[¶] Emmanuel Thomé[†] Luke Valenta[‡]
Benjamin VanderSloot[¶] Eric Wustrow[¶] Santiago Zanella-Béguelin[||] Paul Zimmermann[†]
[*]INRIA Paris-Rocquencourt [†]INRIA Nancy-Grand Est, CNRS, and Université de Lorraine
[||]Microsoft Research [‡]University of Pennsylvania [§]Johns Hopkins [¶]University of Michigan

For additional materials and contact information, visit WeakDH.org.

ABSTRACT

We investigate the security of Diffie-Hellman key exchange as used in popular Internet protocols and find it to be less secure than widely believed. First, we present Logjam, a novel flaw in TLS that lets a man-in-the-middle downgrade connections to "export-grade" Diffie-Hellman. To carry out this attack, we implement the number field sieve discrete log algorithm. After a week-long precomputation for a specified 512-bit group, we can compute arbitrary discrete logs in that group in about a minute. We find that 82% of vulnerable servers use a single 512-bit group, allowing us to compromise connections to 7% of Alexa Top Million HTTPS sites. In response, major browsers are being changed to reject short groups.

We go on to consider Diffie-Hellman with 768- and 1024-bit groups. We estimate that even in the 1024-bit case, the computations are plausible given nation-state resources. A small number of fixed or standardized groups are used by millions of servers; performing precomputation for a single 1024-bit group would allow passive eavesdropping on 18% of popular HTTPS sites, and a second group would allow decryption of traffic to 66% of IPsec VPNs and 26% of SSH servers. A close reading of published NSA leaks shows that the agency's attacks on VPNs are consistent with having achieved such a break. We conclude that moving to stronger key exchange methods should be a priority for the Internet community.