CSE 484 / CSE M 584: Computer Security and Privacy

Cryptography:
Asymmetric Cryptography (finish)

Spring 2016

Franziska (Franzi) Roesner
franzi@cs.washington.edu

Thanks to Dan Boneh, Dieter Gollmann, Dan Halperin, Yoshi Kohno, John Manferdelli, John Mitchell, Vitaly Shmatikov, Bennet Yee, and many others for sample slides and materials...
**Diffie-Hellman Protocol (1976)**

- Alice and Bob never met and share no secrets
- **Public info:** $p$ and $g$
  - $p$ is a large prime number, $g$ is a generator of $\mathbb{Z}_p^*$
    - $\mathbb{Z}_p^* = \{1, 2 \ldots p-1\}$; $\forall a \in \mathbb{Z}_p^* \exists i$ such that $a = g^i \mod p$
    - **Modular arithmetic:** numbers “wrap around” after they reach $p$

---

**Protocol Steps:**

**Alice:**
1. Pick a secret, random $X$
2. Compute $g^x \mod p$

**Bob:**
1. Pick a secret, random $Y$
2. Compute $g^y \mod p$

**Communication:**
- Alice sends $g^x \mod p$ to Bob
- Bob sends $g^y \mod p$ to Alice

**Compute $k$:**
- Alice computes $k = (g^y)^x = g^{xy} \mod p$
- Bob computes $k = (g^x)^y = g^{xy} \mod p$
Diffie-Hellman: Conceptually

Common paint: $p$ and $g$

Secret colors: $x$ and $y$

Send over public transport:
$g^x \mod p$
$g^y \mod p$

Common secret: $g^{xy} \mod p$
Why is Diffie-Hellman Secure?

• Discrete Logarithm (DL) problem: given $g^x \mod p$, it’s hard to extract $x$
  – There is no known efficient algorithm for doing this
  – This is not enough for Diffie-Hellman to be secure!

• Computational Diffie-Hellman (CDH) problem: given $g^x$ and $g^y$, it’s hard to compute $g^{xy} \mod p$
  – ... unless you know $x$ or $y$, in which case it’s easy

• Decisional Diffie-Hellman (DDH) problem: given $g^x$ and $g^y$, it’s hard to tell the difference between $g^{xy} \mod p$ and $g^r \mod p$ where $r$ is random
Properties of Diffie-Hellman

• Assuming DDH problem is hard (depends on choice of parameters!), Diffie-Hellman protocol is a secure key establishment protocol against passive attackers
  – Eavesdropper can’t tell the difference between the established key and a random value
  – Can use the new key for symmetric cryptography

• Diffie-Hellman protocol (by itself) does not provide authentication
Requirements for Public Key Encryption

- **Key generation:** computationally easy to generate a pair (public key $PK$, private key $SK$)
- **Encryption:** given plaintext $M$ and public key $PK$, easy to compute ciphertext $C=E_{PK}(M)$
- **Decryption:** given ciphertext $C=E_{PK}(M)$ and private key $SK$, easy to compute plaintext $M$
  - Infeasible to learn anything about $M$ from $C$ without $SK$
  - Trapdoor function: $\text{Decrypt}(SK,\text{Encrypt}(PK,M))=M$
Some Number Theory Facts

• Euler totient function $\varphi(n)$ ($n \geq 1$) is the number of integers in the $[1,n]$ interval that are relatively prime to $n$
  – Two numbers are relatively prime if their greatest common divisor (gcd) is 1
  – Easy to compute for primes: $\varphi(p) = p-1$
  – Note that $\varphi(ab) = \varphi(a) \varphi(b)$
Some Number Theory Facts

• Euler totient function $\varphi(n)$ ($n \geq 1$) is the number of integers in the [1,n] interval that are relatively prime to n
  – Two numbers are relatively prime if their greatest common divisor (gcd) is 1
  – Easy to compute for primes: $\varphi(p) = p-1$
  – Note that $\varphi(ab) = \varphi(a) \varphi(b)$

• Euler’s theorem: if $a \in Z_n^*$, then $a^{\varphi(n)} = 1 \mod n$
  $Z_n^*$: integers relatively prime to n
RSA Cryptosystem [Rivest, Shamir, Adleman 1977]

• **Key generation:**
  – Generate large primes $p$, $q$
    • Say, 1024 bits each (need primality testing, too)
  – Compute $n = pq$ and $\varphi(n) = (p-1)(q-1)$
  – Choose small $e$, relatively prime to $\varphi(n)$
    • Typically, $e = 3$ or $e = 2^{16}+1 = 65537$
  – Compute unique $d$ such that $ed = 1 \mod \varphi(n)$
    • Modular inverse: $d = e^{-1} \mod \varphi(n)$
  – Public key = $(e, n)$; private key = $(d, n)$

• **Encryption** of $m$: $c = m^e \mod n$

• **Decryption** of $c$: $c^d \mod n = (m^e)^d \mod n = m$
Why RSA Decryption Works

e \cdot d \equiv 1 \pmod{\varphi(n)}, \text{ thus } e \cdot d = 1 + k \cdot \varphi(n) \text{ for some } k

Let \(m\) be any integer in \(\mathbb{Z}_n^*\) (not all of \(\mathbb{Z}_n\))

\[ cd \mod{n} = (m^e)^d \mod{n} = m^{1+k \cdot \varphi(n)} \mod{n} \]

= \((m \mod{n}) \cdot (m^{k \cdot \varphi(n)} \mod{n})\)

Recall: Euler’s theorem: if \(a \in \mathbb{Z}_n^*\), then \(a^{\varphi(n)} \equiv 1 \pmod{n}\)

\[ cd \mod{n} = (m \mod{n}) \cdot (1 \mod{n}) \]

= \(m \mod{n}\)

Proof omitted: True for all \(m\) in \(\mathbb{Z}_n\), not just \(m\) in \(\mathbb{Z}_n^*\)
Why is RSA Secure?

• **RSA problem:** given $c$, $n=pq$, and $e$ such that $\gcd(e, \varphi(n))=1$, find $m$ such that $m^e = c \mod n$
  
  – In other words, recover $m$ from ciphertext $c$ and public key $(n,e)$ by taking $e^\text{th}$ root of $c$ modulo $n$
  
  – There is no known efficient algorithm for doing this

• **Factoring problem:** given positive integer $n$, find primes $p_1, \ldots, p_k$ such that $n=p_1^{e_1} p_2^{e_2} \ldots p_k^{e_k}$

• If factoring is easy, then RSA problem is easy (knowing factors means you can compute $d = \text{inverse of } e \mod (p-1)(q-1)$)
  
  – It may be possible to break RSA without factoring $n$ -- but if it is, we don’t know how
RSA Encryption Caveats

• Encrypted message needs to be interpreted as an integer less than n
• Don’t use RSA **directly** for privacy – **output is deterministic!** Need to pre-process input somehow
• Plain RSA also does **not** provide integrity
  – Can tamper with encrypted messages

In practice, OAEP is used: instead of encrypting $M$, encrypt $M \oplus G(r) \oplus H(M \oplus G(r))$
  – $r$ is random and fresh, $G$ and $H$ are hash functions
Digital Signatures: Basic Idea

Given:

- Everybody knows Bob’s **public key**
- Only Bob knows the corresponding **private key**

Goal:

- Bob sends a “digitally signed” message
  1. To compute a signature, must know the private key
  2. To verify a signature, only the public key is needed
RSA Signatures

• Public key is \((n,e)\), private key is \((n,d)\)

• To sign message \(m\): \(s = \text{m^d mod n}\)
  – Signing & decryption are same underlying operation in RSA
  – It’s infeasible to compute \(s\) on \(m\) if you don’t know \(d\)

• To verify signature \(s\) on message \(m\):
  verify that \(s^e \text{ mod n} = (\text{m^d)^e mod n} = m\)
  – Just like encryption (for RSA primitive)
  – Anyone who knows \(n\) and \(e\) (public key) can verify signatures produced with \(d\) (private key)

• In practice, also need padding & hashing
  – Standard padding/hashing schemes exist for RSA signatures
DSS Signatures

- Digital Signature Standard (DSS)
- Public key: \((p, q, g, y = g^x \mod p)\), private key: \(x\)
- Security of DSS requires hardness of discrete log
  - If could solve discrete logarithm problem, would extract \(x\) (private key) from \(g^x \mod p\) (public key)
Advantages of Public Key Crypto

• Confidentiality without shared secrets
  – Very useful in open environments
  – Can use this for key establishment, with fewer “chicken-or-egg” problems
    • With symmetric crypto, two parties must share a secret before they can exchange secret messages

• Authentication without shared secrets
  – Use digital signatures to prove the origin of messages

• Encryption keys are public, but must be sure that Alice’s public key is really her public key
  – This is a hard problem...
Disadvantages of Public Key Crypto

• Calculations are 2-3 orders of magnitude slower
  – Modular exponentiation is an expensive computation
  – Typical usage: use public-key cryptography to establish a shared secret, then switch to symmetric crypto
    • E.g., IPsec, SSL, SSH, ...

• Keys are longer
  – 1024+ bits (RSA) rather than 128 bits (AES)

• Relies on unproven number-theoretic assumptions
  – What if factoring is easy?
    • Factoring is believed to be neither P, nor NP-complete
  – (Of course, symmetric crypto also rests on unproven assumptions...)
Problem: How does Alice know that the public key she received is really Bob’s public key?
Threat: Man-In-The-Middle (MITM)
Distribution of Public Keys

• Public announcement or public directory
  – Risks: forgery and tampering

• Public-key certificate
  – Signed statement specifying the key and identity
    • $\text{sig}_{\text{CA}}(\text{“Bob”}, \text{PK}_B)$

• Common approach: certificate authority (CA)
  – Single agency responsible for certifying public keys
  – After generating a private/public key pair, user proves his identity and knowledge of the private key to obtain CA’s certificate for the public key (offline)
  – Every computer is pre-configured with CA’s public key