CSE 484 / CSE M 584: Computer Security and Privacy

Cryptography
[Asymmetric Cryptography]

Autumn 2020

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

Thanks to Dan Boneh, Dieter Gollmann, Dan Halperin, Yoshi Kohno, Ada Lerner, John Manferdelli, John Mitchell, Vitaly Shmatikov, Bennet Yee, and many others for sample slides and materials ...
**Diffie-Hellman Key Exchange**

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

### Alice and Bob Protocol

1. **Pick secret, random $X$**
2. **Pick secret, random $Y$**
3. **Compute $k=(g^y)^x = g^{xy} \mod p$**
4. **Compute $k=(g^x)^y = g^{xy} \mod p$**
Example Diffie Hellman Computation

Public

\[ p = 11, \quad g = 2 \]

Alice

\[ x = 9 \]

Bob

\[ y = 4 \]

\[ g^x \mod p \]

\[ 2^9 \mod 11 \]

\[ g^y \mod p \]

\[ 2^4 \mod 11 \]

\[ g^{xy} \mod p \]

\[ 2^{9 \cdot 4} \mod 11 = 9 \]

\[ \text{hash}(9) = K \]

\[ \text{asc123} = \text{AES key} \]
Why is Diffie-Hellman Secure?

\[ 5 = 2^0 \mod 11 \]

- **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
**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:** $g^{xy} \mod p$

[from Wikipedia]
Diffie-Hellman Caveats

• Assuming DDH problem is hard (depends on choice of parameters!), Diffie-Hellman protocol is a secure key establishment protocol against passive attackers
  – Common recommendation:
    • Choose $p=2q+1$, where $q$ is also a large prime
    • Choose $g$ that generates a subgroup of order $q$ in $\mathbb{Z}_p^*$
  – Eavesdropper can’t tell the difference between the established key and a random value
  – In practice, often hash $g^{xy} \mod p$, and use the hash as the key
  – Can use the new key for symmetric cryptography

• Diffie-Hellman protocol (by itself) does not provide authentication (against active attackers)
  – Person in the middle attack (also called “man in the middle attack”)
Person In The Middle Attack

$g, p$

$A \xrightarrow{g^x \mod p} M$ \text{Mallory} \xleftarrow{g^m \mod p} B

"I'm Alice!"

$g^y \mod p$
More on Diffie-Hellman Key Exchange

• Important Note:
  – We have discussed discrete logs modulo integers
  – Significant advantages in using elliptic curve groups
    • Groups with some similar mathematical properties (i.e., are “groups”) but have better security and performance (size) properties
Stepping Back: Asymmetric Crypto

• We’ve just seen **session key establishment**
  – Can then use shared key for symmetric crypto

• Next: **public key encryption**
  – For confidentiality

• Then: **digital signatures**
  – For authenticity

\[
\text{Encrypt}(\text{Msg}, B_{\text{pub}}) = C
\]

\[
\text{Decrypt}(C, B_{\text{priv}}) = \text{Msg}
\]
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 $\varphi(ab) = \varphi(a) \varphi(b)$ if $a$ & $b$ are relatively prime
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 \( \Phi(n) = (p-1)(q-1) \)
  – Choose small e, relatively prime to \( \Phi(n) \)
    • Typically, \( e = 3 \) or \( e = 2^{16} + 1 = 65537 \)
  – Compute unique d such that \( ed \equiv 1 \mod \Phi(n) \)
    • Modular inverse: \( d \equiv e^{-1} \mod \Phi(n) \)
  – 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 (FYI)

\[ e \cdot d = 1 \mod \phi(n), \text{ thus } e \cdot d = 1+k \cdot \phi(n) \text{ for some } k \]

Let \( m \) be any integer in \( Z_n^* \) (not all of \( Z_n \))

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

Recall: Euler’s thm: if \( a \) in \( Z_n^* \), then \( a^{\phi(n)} = 1 \mod n \)

\[ c^d \mod n = (m \mod n) \cdot (1 \mod n) \]
\[ = m \mod n \]

Proof omitted: True for all \( m \) in \( Z_n \), not just \( m \) in \( Z_n^* \)
Why RSA Decryption Works (FYI)

• Decryption of c: \( c^d \mod n = (m^e \mod n)^d \mod n = (m^e)^d \mod n = m \)
• Recall \( n=pq \) and \( \varphi(n)=(p-1)(q-1) \) and \( ed \equiv 1 \mod \varphi(n) \)

• Chinese Remainder Theorem: To show \( m^ed \mod n \equiv m \mod n \), sufficient to show:
  – \( m^ed \mod p \equiv m \mod p \)
  – \( m^ed \mod q \equiv m \mod q \)

• If \( m \equiv 0 \mod p \) \( \rightarrow \) \( m^ed \equiv 0 \mod p \)

• Else \( m^ed = m^{ed-1}m = m^{k(q-1)(p-1)}m = m^{h(p-1)}m \) for some \( k \), and \( h=k(q-1) \). Why? Recall how \( d \) was chosen and the definition of mod.
• Fermat Little Theorem: \( m^{(p-1)h}m \equiv 1^h m \mod p \equiv m \mod p \)
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 \( e \)-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 \( n = p_1^{e_1} p_2^{e_2} \cdots p_k^{e_k} \)

- 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 $M \oplus G(r); r\oplus H(M \oplus G(r))$
  – 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 \(m\):  
  \[ s = m^d \mod n \]
  – 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 \(s^e \mod n = (m^d)^e \mod n = m\)
  – 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)

• Again: We’ve discussed discrete logs modulo integers;
significant advantages to using elliptic curve groups instead.
Cryptography Summary

• Goal: Privacy
  – Symmetric keys:
    • One-time pad, Stream ciphers
    • Block ciphers (e.g., DES, AES) → modes: EBC, CBC, CTR
  – Public key crypto (e.g., Diffie-Hellman, RSA)

• Goal: Integrity
  – MACs, often using hash functions (e.g., SHA-256)

• Goal: Privacy and Integrity
  – Encrypt-then-MAC

• Goal: Authenticity (and Integrity)
  – Digital signatures (e.g., RSA, DSS)
Want More Crypto?

• Some suggestions:
  – CSE 490C (Rachel Lin):
    https://courses.cs.washington.edu/courses/cse490c/20au/
  – Stanford Coursera (Dan Boneh):
    https://www.coursera.org/learn/crypto