Lecture 13: Public Key Cryptography Part 2

COSC362 Data and Network Security

Book 1: Chapters 9 and 10 – Book 2: Chapters 2 and 21

Spring Semester, 2021

Motivation Reminder

- Public key cryptography (PKC) has features that symmetric key cryptography does not have.
- Applied for key management in protocols such as TLS and IPsec.
- RSA is one of the best known public key cryptosystems, widely deployed in practice.
- Alternatives include discrete log based ciphers, also widely deployed and standardised.

Outline

Diffie-Hellman Key Exchange
- Protocol
- Properties

Elgamal Cryptosystem
- Algorithms
- Security

Elliptic Curves

Recent Developments

Outline

Diffie-Hellman Key Exchange
- Protocol
- Properties

Elgamal Cryptosystem
- Algorithms
- Security

Elliptic Curves

Recent Developments

Diffie-Hellman Key Exchange

- 2 users, Alice and Bob, share a secret using only public communication.
- Public elements:
  - Large prime p
  - Generator $g \in \mathbb{Z}_p^*$
- Alice and Bob each selects random values a and b respectively, where $1 < a, b < p$:
  - Alice sends g^a to Bob over an *insecure* channel.
  - Bob sends g^b to Alice over an *insecure* channel.
- Alice and Bob both compute the secret key $Z = g^{ab} \mod p$.

Protocol

Alice		Bob
Choose a		
	$\xrightarrow{g^a \bmod p}$	
		Choose b
	$\xleftarrow{g^b \bmod p}$	
$Z = (g^b)^a \bmod p$		$Z = (g^a)^b \bmod p$

Z can be used to compute a key (e.g. AES) by using a *key derivation function* based on a public hash function.

Example

Public elements are $p = 181$ and $g = 2$.

- Selecting private keys:
  - Alice selects $a = 50$
  - Bob selects $b = 33$
- Sharing public keys:
  - Alice sends $g^a \mod p = 2^{50} \mod 181 \equiv 116$ to Bob
  - Bob sends $g^b \mod p = 2^{33} \mod 181 \equiv 30$ to Alice
- Computing the shared key:
  - Alice computes $Z = (g^b)^a \mod p \equiv 30^{50} \mod 181$
  - Bob computes $Z = (g^a)^b \mod p \equiv 116^{33} \mod 181$

The common secret is $Z = 49$.

Security

- An attacker who finds discrete logarithms breaks the protocol:
  - Intercepting $g^a \mod p$ and taking the discrete log to get a.
  - Computing $(g^b)^a$ in the same way as Bob.
- No better way known for a passive adversary than by taking discrete logs:
  - It is unknown if there is a better way.

Authenticated Diffie-Hellman

- In the basic protocol:
  - Messages between Alice and Bob are not authenticated.
- In a network, Alice/Bob do not know how Z is shared, unless messages are authenticated.
- Man-in-the-middle attack: the adversary sets up 2 keys, 1 with Alice and 1 with Bob, and relays messages between the 2.
- Authentication feature:
  - Authentication can be added by using digital signatures.

Authenticated Diffie-Hellman

Alice		Bob
Choose a		
	$\xrightarrow{A,\, g^a \bmod p}$	
		Choose b
	$\xleftarrow{B,\, g^b \bmod p,\, Sig_B(B, A, g^b)}$	
	$\xrightarrow{Sig_A(A, B, g^a)}$	
$Z = (g^b)^a \bmod p$		$Z = (g^a)^b \bmod p$

- Signature $Sig_A(m)$ on message m by Alice
- Signature $Sig_B(m)$ on message m by Bob
- Both parties know each other's public signature verification key.

Static and Ephemeral Diffie-Hellman

- The above protocol uses *ephemeral keys*:
  - Key used once and then discarded.
- In the *static* protocol:
  - Alice chooses a long-term private key x_A and public key $y_A = g^{x_A} \mod p$.
  - Bob chooses a long-term private key x_B and public key $y_B = g^{x_B} \mod p$.
- Alice and Bob find a shared secret $S = g^{x_A x_B} \mod p$, that is static:
  - S stays the same until Alice and Bob change their public keys.

Outline

Diffie-Hellman Key Exchange
- Protocol
- Properties

Elgamal Cryptosystem
- Algorithms
- Security

Elliptic Curves

Recent Developments

Elgamal Cryptosystem

- Proposed by Taher Elgamal in 1985.
- Diffie-Hellman protocol turned into a cryptosystem.
- For encryption and for signature.
- Alice combines her ephemeral private key with Bob's long-term public key.

Key Generation

Key generation:

- Select a prime p and a generator $g \in \mathbb{Z}_p^*$.
- Select a long-term private key $K_D = x$, where $1 < x < p$.
- Compute $y = g^x \mod p$.
- Set the long-term public key as $K_E = (p, g, y)$.

Encryption and Decryption

Encryption:

- $K_E = (p, g, y)$ is the public key for encryption.
- Select a message M where $0 < M < p$.
- Choose at random an ephemeral private key k.
- Compute $g^k \mod p$ and $My^k \mod p$.
- Set the ciphertext as:

$$C = (C_1, C_2) = Enc(M, K_E) = (g^k \mod p, My^k \mod p)$$

Decryption:

- $K_D = x$ is the private key for decryption.
- $C = (C_1, C_2)$ is the ciphertext.
- Compute $C_1^x \mod p$.
- $Dec(C, K_D) = C_2 \cdot (C_1^x)^{-1} \mod p = M$.

Correctness

- Alice knows the ephemeral private key k.
- Bob knows the static/long-term private key $K_D = x$.
- Both Alice and Bob compute the Diffie-Hellman value for the 2 public keys:
  - $C_1 = g^k \mod p$
  - $y = g^x \mod p$
- Diffie-Hellman value $y^k \mod p = C_1^x \mod p$ used as a mask for the message M.

Example

Key generation:

- Choose prime $p = 181$ and generator $g = 2$.
- Bob's private key is $x = 50$.
- Compute $y = g^x \mod p = 116$.
- Bob's public key is $(181, 2, 116)$.

Encryption:

- Alice wants to send $M = 97$.
- Alice chooses at random $k = 31$.
- Ciphertext is $C = (C_1, C_2) = (98, 173)$.

Decryption:

- Bob receives $C = (C_1, C_2)$.
- Bob computes $C_1^x \mod p = 98^{50} \mod 181 = 138$.
- Bob recovers $M = C_2 \times (C_1^x)^{-1} \mod p = 173 \times 138^{-1} \mod 181 = 97$.

Security

- An attacker who solves the discrete log problem breaks Elgamal cryptosystem by determining the private key x from $g^x \mod p$.
- Possible for many users to share the same p and g.
- No need for any padding as in RSA:
  - Each ciphertext is already randomised, thanks to the ephemeral key k.
- Security proof in a suitable model, subject to the difficulty of *decisional Diffie-Hellman problem*.

Outline

Diffie-Hellman Key Exchange
- Protocol
- Properties

Elgamal Cryptosystem
- Algorithms
- Security

Elliptic Curves

Recent Developments

Elliptic Curves

- Algebraic structures formed from cubic equations.
- Curves defined over any field.
- Example:
  - Set of all (x, y) pairs which satisfy $y^2 = x^3 + ax + b \mod p$.
  - Curve over field $\mathbb{Z}_p$.
- Add an identity element, and then define a binary operation on the points (e.g. multiplication):
  - Form a group over the elliptic curve points, called *elliptic curve group*.

Choosing Elliptic Curves

- Generate a new elliptic curve at any time:
  - But applications usually use standardised curves.
  - Standard: FIPS 186-4 (NIST curves, 2013).
- Standardised curves generated in a verifiably random way:
  - Difficult to generate curves with any hidden special properties.

Example

NIST curve P-192:

- Curve of n points over $\mathbb{Z}_p$ with generator (G_x, G_y) and equation $y^2 = x^3 - 3x + b \bmod p$.
- p and n are 192 bits long.
- s is the seed for random generation.
- c is the output of a SHA-1 hash generated from s.

$$
\begin{aligned}
p &= 6277101735386680763835789423207666416083908700390324961279 \\
n &= 6277101735386680763835789423176059013767194773182842284081 \\
s &= \texttt{3045ae6fc8422f64ed579528d38120eae12196d5} \\
c &= \texttt{3099d2bbbfcb2538542dcd5fb078b6ef5f3d6fe2c745de65} \\
b &= \texttt{64210519e59c80e70fa7e9ab72243049feb8deecc146b9b1} \\
G_x &= \texttt{188da80eb03090f67cbf20eb43a18800f4ff0afd82ff1012} \\
G_y &= \texttt{07192b95ffc8da78631011ed6b24cdd573f977a11e794811}
\end{aligned}
$$

Discrete Logarithm

- Discrete log defined on elliptic curve groups:
  - If elliptic curve operation denoted as multiplication, then definition same as in $\mathbb{Z}_p^*$.
- Best known algorithms for solving discrete log problem are *exponential* in the length of parameters.
- Elliptic curve implementations use smaller keys.
- Comparison with RSA:
  - Relative advantage of elliptic curve cryptography increases at higher security levels.

Security Comparison

Sym. key length	RSA modulus length	EC element length
80	1024	160
128	3072	256
192	7680	384
256	15360	512

Example: brute force search of 128-bit key for AES takes roughly same computational effort as factorisation of 3072-bit RSA modulus or for taking discrete logarithms in an elliptic curve with elements of size 256 bits.

Standard: NIST SP 800-57 Part 1 Recommendations for Key Management (revised 2016).

Elliptic Curve Cryptography

- Most cryptosystems based on discrete log constructed with elliptic curves as well as in $\mathbb{Z}_p^*$.
- Examples of cryptosystems run on elliptic curves:
  - Diffie-Hellman key exchange
  - Elgamal encryption
- Canadian company Certicom holds ECC-based patents:
  - `https://www.certicom.com/content/certicom/en/about.html`

Outline

Diffie-Hellman Key Exchange
Protocol
Properties

Elgamal Cryptosystem
Algorithms
Security

Elliptic Curves

Recent Developments

Identity-Based Cryptography

- Proposed by Shamir (1982).
- Extensively researched in the 2000s, with the use of elliptic curve *pairings*.
- Public keys and certificates are not needed:
  - Identity of the key owner replaces the public key.
  - Message encryption using public parameters and recipient's identity.
- Limitation:
  - Need of a trusted key generation process.
- Generalisation with functional cryptography:
  - General access policies used to define who may decrypt the ciphertext.

Post-Quantum Cryptography

- Most current public key cryptography will be broken if quantum computers become available:
  - Shor's algorithm enabling factorisation.
  - Shor's algorithm also enabling to find discrete logarithms.
- Concerns: building cryptographic primitives still secure if this happens!
- Symmetric key cryptography can be used but with double-length keys:
  - Grover's algorithm allowing searching.
- Post-quantum cryptosystems based on different problems:
  - Lattice problems, coding theory, multi-variable polynomial resolution.
- NIST process to standardise is under way:
  - `https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization`