University Heidelberg
Faculty Mathematikon
Seminar: Prime numbers and Cryptography with supervisor Dr. Barinder Banwait
Burhan Akin Yilmaz
April, 2022

# 1 Key-Exchange

Following scheme allows two parties to exchange a secret-key even under passive attacks like eavesdropping. A KE-protocol over $\mathcal{K}$ consists of two interactive probabilistic-polytime-algorithms KE = ($\text{KE}_A$, $\text{KE}_B$) which output a key $\text{sk}_A$ and $\text{sk}_B$. We want perfect correctness, such that those algorithm always agree to the same key $\text{sk}_A = \text{sk}_B$.

The security of such algorithms is defined over a game:

- $\text{KE}_A$ and $\text{KE}_B$ interact with each other, agree to $\text{sk}_A = \text{sk}_B$ and store all exchanged messages in $\tau$.
- Our attacker will try to output $\text{sk}^* \leftarrow \mathcal{A}(\tau)$ such that $\text{sk}^* = \text{sk}_A$.

We want the attacker to only have negligible chances to succeed in this game.

## 1.1 Discrete Log and Computational Diffie-Hellman assumption

Let $\mathbb{Z}_p^*$ be any cyclic group of order $p-1$.

- The discrete logarithm is assumed to be a hard problem. Given $h = g^x \mod p$ and generator $g$, find smallest exponent x.
- The computational diffie-hellman assumption. Let $x \xleftarrow{\$} \mathbb{Z}_p$ and $y \xleftarrow{\$} \mathbb{Z}_p$. Determining $g^{xy}$ given $(\mathbb{Z}_p^*, p, g, g^x, g^y)$ is computational infeasible.

## 1.2 Diffie-Hellman Key-Exchange

For our Let $p$ be a prime number, and $\mathbb{Z}_p^*$ be a cyclic group of order $p-1$. Furthemore $\mathbb{Z}_p^* = \langle g \rangle$. Consider following protocol to exchange keys.

| $\text{KE}_A$ | | $\text{KE}_B$ |
|---|---|---|
| $x \xleftarrow{\$} \mathbb{Z}_p$ | | |
| | $\xrightarrow{g^x}$ | |
| | $\xleftarrow{g^y}$ | $y \xleftarrow{\$} \mathbb{Z}_p$ |
| **return** $(g^y)^x$ | | **return** $(g^x)^y$ |

# 2 RSA Cryptosystems

The RSA encryption scheme works very simple and is based on the difficulty of factorization and the RSA-assumption. First pick two large prime numbers $p, q$ and calculate our RSA-modulus $n = pq$. Next you determine two integers $e, s \geq 3$, such that $es \equiv 1 \mod (p-1)(q-1)$. Here we need to pick $e$ coprime to $\phi(n) = (p-1)(q-1)$, only then a solution can exist. So $gcd(e, \phi(n)) = 1$. Determine s with the extended euclidean algorithm $extended\_gcd(e, (p-1)(q-1))$. As public key use $(n, e)$, as private key $(n, s)$ and encrypt messages with $y = x^e \mod n$. Our $y$ is our ciphertext, and we decrypt with $y^s \equiv x \mod n$, and if our plainttext $x$ was used from the interval $\{0, ..., n-1\}$ then we have $y^s = x \mod n$.

# 3 Digital Signatures

So far we have Alice and Bob communicating securely over an in-secure channel in the sense of confidentality. But we have no integrity and authenticity so far. With a digital signature we assure authenticity and integrity of a message. A digital signature over $(\mathcal{K}_{sk}, \mathcal{K}_{pub}, \mathcal{M}, \mathcal{S})$ is a tuple SIG = (Gen, Sign, Vfy) of PPT-algorithms.

- Gen() will generate our public verification key and secret key $(vk, sk) \in \mathcal{K}_{pub} \times \mathcal{K}_{sk}$.
- Sign$(sk, x) \rightarrow s$ will generate our signature for our message x.
- Vfy$(vk, x, s) \rightarrow \{0, 1\}$ is a deterministic algorithm that outputs 1 if the signature was generated over the message x with the secret key.

One can apply RSA to construct a digital signature.Alice and Bob generate their own public and secret keys, $sk_A = (n_A, s_A), vk_A = (n_A, e_A)$ and $sk_B = (n_B, s_B), vk_B = (n_B, e_B)$. We now assume, that the public keys are known to each other. First Bob creates his signature $s = x^{s_B} \mod n_B$ and sends this signature to Alice using her public key $y = s^{e_A} \mod n_A$. Alice now decrypts $s = y^{s_A} \mod n_A$ and can verify this signature belongs to message x and is authenticated by Bob by $x \stackrel{?}{=} s^{e_B} \mod n_B$.

| *Alice* | | *Bob* |
|---|---|---|
| $x, x \stackrel{?}{=} s^{e_B} \mod n_B$ | $\longleftarrow$ $x, y = s^{e_A} \mod n_A$ | $x, s = x^{s_B} \mod n_B$ |

# 4 Elliptic Curve Cryptosystems

Given an elliptic curve E over $\mathbb{F}_p$ is an equation

$$y^2 = x^3 + ax + b,$$

where $a, b \in \mathbb{F}_p$ and $4a^3 + 27 \neq 0$. We know that the points on the elliptic curve define with the addition operator $\boxplus$ a group.

Now we can define our earlier assumptions and protocols on this elliptic curve. For the elliptic discrete logarithm we choose a point of prime order $q$, such that $qP = \mathcal{O}$. Now we give an adversary our point $P$ and our $\alpha P$ with $\alpha \in \mathbb{Z}_q$. Determining $\alpha$ is considered to be a hard problem. The operation $\alpha P := (\alpha - 1)P \boxplus P$ can be computed with at most $2 \log_2 \alpha$ operations (fast exponentiation using our $\alpha$). On the other hand, the best known algorithm to solve the discrete logarithm over elliptic curves has time complexity $\mathcal{O}(\sqrt{n})$.

# 5 Elliptic Curve Diffie Hellman (ECDH)

| KE$_A$ | | KE$_B$ |
|---|---|---|
| $K_A \xleftarrow{\$} \{2, ..., n-2\}$ | | |
| | $\xrightarrow{Q = K_A P}$ | |
| | $\xleftarrow{R = K_B P}$ | $K_B \xleftarrow{\$} \{2, ..., n-2\}$ |
| **return** $K_B Q$ | | **return** $K_A Q$ |

# 6 Elliptic Curve Digital Signature Algorithm (ECDSA)

Consider following digital Signature using elliptic curves [CP05]. Alice wants to sign a message $x$ and Bob verifies it.

- Gen()

  Step 1: Alice chooses a curve with $|E(\mathbb{F}_p)| = fr$. Finds a point of prime order $r$.

  Step 2: Now she chooses a random integer $d \in [2, r-2]$.

  Step 3: Gen will **return**$((E, P, r, Q), d)$.

- Sign$(d, x)$

  Step 1: Alice chooses a random $k \in [2, r-2]$.

  Step 2: $(x_1, y_1) = kP$

  Step 3: $R = x_1 \mod r$

  Step 4: $s = k^{-1}(h(x) + Rd) \mod r$

  Step 5: if $s == 0$ goto Sign$(x)$

  Step 6: **return** $(R, s)||x$

- Verify$((E, P, r, Q), x)$

  Step 1: $w = s^{-1} \mod r$

  Step 2: $u_1 = h(x)w \mod r$

  Step 3: $u_2 = Rw \mod r$

  Step 4: $(x_0, y_0)$

  Step 5: $v = x_0 \mod r$

  Step 6: if $v == R$ **return** 1 else **return** 0

# 7 Coin-Flip Protocol

A commitment scheme has two properties:

- **Hiding**: You cannot conclude the actual bit $b$ committed from $c$.
- **Binding**: When commiting a bit $b$, you can not send an opening string such that a different bit $\overline{b}$ is opened.

The coin-flip protocol can be implemented using different elegant ideas like Naos construction with pseudorandom generators [90] or with number theoretical approaches using congruences with primes. The latter one is of our interest.

Step 1: Alice computes two distinct random primes $p, q$, calculates $n = pq$, and finds a random prime $r$ such that $n$ is quadratic nonresidue $\mod p$, resp. $(\frac{n}{r}) = -1$.

Step 2: Alice sends Bob the commitment string $(n, r)$.

Step 3: Bob sends Alice his guess of which of the prime factors of $n$ is a quadratic residue.

Step 4: Alice sends the opening string $(p, q)$.

Obviously the binding property is satisfied, since $n$ has a unique factorization $n = pq$.

# References

[90] *Bit Commitment Using Pseudo-Randomness **. 1990. URL: https://link.springer.com/content/pdf/10.1007/0-387-34805-0_13.pdf.

[CP05] Richard Crandall and Carl Pomerance. *Prime Numbers - A Computational Perspective*. 2005.

[Sho20] Boneh Shoup. *A Graduate Course in Applied Cryptography*. 2020. URL: http://toc.cryptobook.us/book.pdf (visited on 04/03/2022).