Cryptography 4: Asymmetric Cryptography

Maria Eichlseder

Information Security – WT 2019/20

You Are Here

Crypto 1	Crypto 2	Crypto 3	Crypto 4
Introduction to InfoSec & Crypto	Symmetric Authentication	Symmetric Encryption	Asymmetric Cryptography
▪ Terminology ▪ Security notions ▪ Keys, Kerckhoffs' principle	➔ Integrity ▪ Hash functions ▪ MACs (Message Authentication)	➔ Confidentiality ▪ AEAD (Auth. Encryption) ▪ Symmetric primitives	➔ Establishing communication ▪ Key exchange ▪ Signatures ▪ Asymmetric primitives

Recap of Last Week (1): Schemes for Encryption

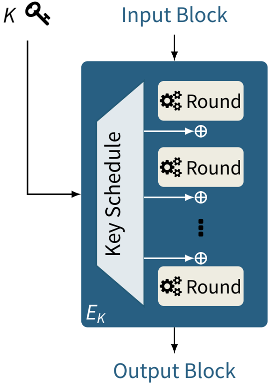
Encryption schemes transform a plaintext Message M of arbitrary length to a Ciphertext C of about the same length based on a Key K of fixed length.

Schemes may accept additional inputs or produce an authentication Tag T.

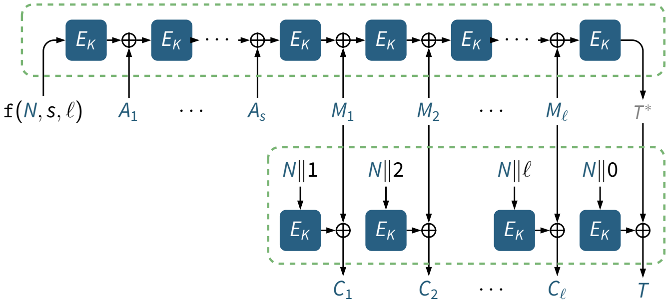
Encryption $\mathcal{E}_{K_{AB}}$	Auth. Enc. $\mathcal{AE}_{K_{AB}}$	Key Encapsulation
Symmetric Key K_{AB}	Symmetric Key K_{AB}	Asymmetric Keys
Confidentiality only	Confid. + Authenticity	Confidentiality
A, B can encrypt	A, B can encrypt + auth	Anyone can encrypt
A, B can decrypt	A, B can decrypt + verify	A can decrypt

Recap of Last Week (2): Layers of the Symmetric Crypto Stack

Primitive (e.g., AES)



Mode of Operation (e.g., AES-CCM)



Outline

Background

- Motivation, Goals, Applications
- Recap: Modular Arithmetic and Hard Problems

Key Exchange

- Diffie–Hellman Key Exchange

Asymmetric Encryption

- Trapdoor One-way Functions
- RSA Public-Key Encryption

Signatures

- RSA Signatures

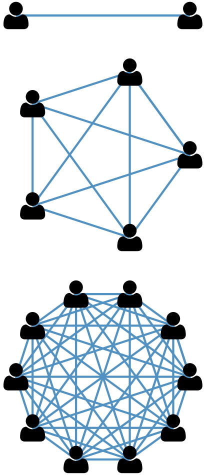
Discussion

- Today's and Tomorrow's Schemes: ECC, PQC, …

Background

Introduction

Limitations of Symmetric Cryptography

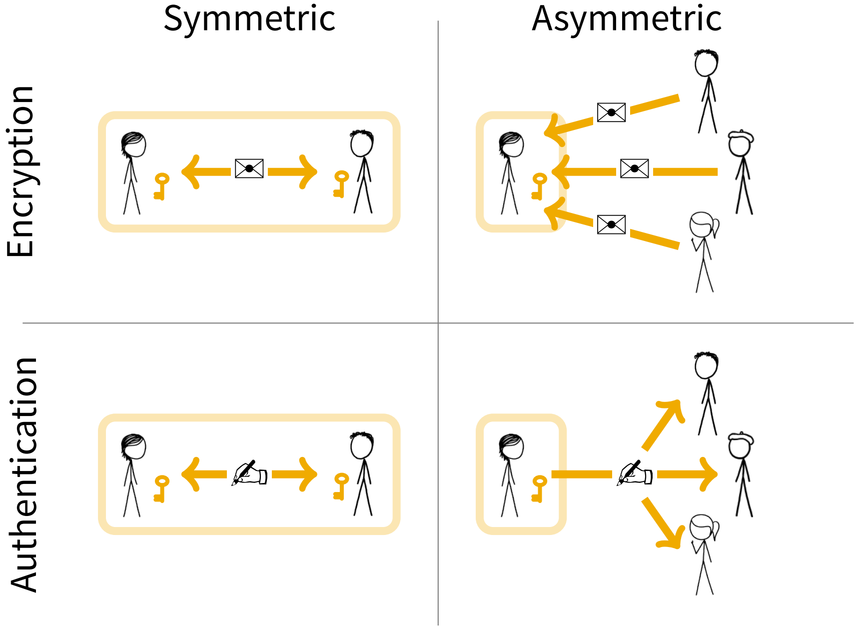


Key Distribution

- System with n users needs $\binom{n}{2} = \frac{n\cdot(n-1)}{2}$ key-pairs
- Adding new users is expensive and complicated
- How would this work for securing the internet?!

Symmetric Trust Relationships

- Assumes that users trust each other equally
- Does not support establishing new connections
- Does not support properties like non-repudiation


Symmetric
Asymmetric
Encryption
Authentication

Asymmetric Crypto Schemes

Key Exchange

Two Keypairs K_A, K_B

A and *B* communicate to agree on a new symmetric key

- *A*, *B* can influence key
- *A*, *B* can derive key

Encryption

Asymmetric Keypair K_A

A receives confidential messages (usually an “encapsulated” key)

- Anyone can encrypt
- *A* can decrypt

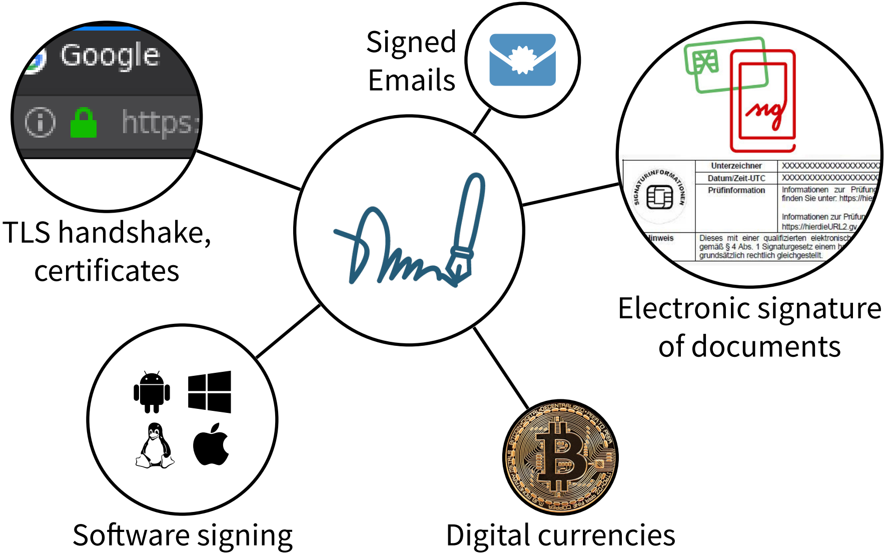
Signature

Asymmetric Keypair K_A

A creates a signature to authenticate messages

- *A* can authenticate
- Everyone can verify

Applications of Digital Signatures



Applications of Key Exchange and Asymmetric Encryption

Key Exchange is used to **agree on a session key** to be used for a symmetrically protected communication channel

- Secure Communication via TLS
- IPsec for protecting VPNs
- SSH Secure Shell
- …

Asymmetric Encryption is mostly used to **send a session key** for a symmetrically protected message (“key encapsulation”)

- SSH Secure Shell
- Email encryption with PGP or S/MIME
- …

Recap: Modular Arithmetic and the Set $\mathbb{Z}_n$

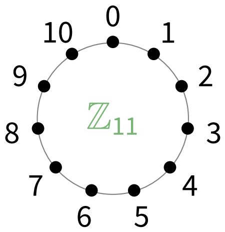
We arrange integers in classes by their remainder after division by the modulus n
(aka “modulo n”, “reduce by n”)

$\mathbb{Z}_n = \{0, \ldots, n-1\}$ is the set of all classes modulo n.

Integers a, b in the same class are “congruent mod n”: “$a \equiv b \pmod{n}$”.

Example: mod 11

Class $\in \mathbb{Z}_{11}$	Integers $\subseteq \mathbb{Z}$
0	$\{\ldots, -11, 0, 11, 22, \ldots\}$
1	$\{\ldots, -10, 1, 12, 23, \ldots\}$
2	$\{\ldots, -9, 2, 13, 24, \ldots\}$
$\vdots$	$\vdots$
10	$\{\ldots, -1, 10, 21, 32, \ldots\}$



Recap: The Additive Group $(\mathbb{Z}_n, +)$

The set $\mathbb{Z}_n$ with the operation $+$ (addition modulo n) is a group that satisfies:

1. Associativity: $\forall a, b, c \in \mathbb{Z}_n : \; a + (b + c) = (a + b) + c$
2. Commutativity: $\forall a, b \in \mathbb{Z}_n : \; a + b = b + a$
3. Neutral element 0: $\forall a \in \mathbb{Z}_n : \; a + 0 = a = 0 + a$
4. Inverse element $-a$ for every element $a \in \mathbb{Z}_n$: $\quad a + (-a) = 0$

Example $(\mathbb{Z}_{11}, +)$:

+	0	1	2	3	4	5	6	7	8	9	10
0	0	1	2	3	4	5	6	7	8	9	10
1	1	2	3	4	5	6	7	8	9	10	0
2	2	3	4	5	6	7	8	9	10	0	1
3	3	4	5	6	7	8	9	10	0	1	2
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
10	10	0	1	2	3	4	5	6	7	8	9

Recap: The Multiplicative Group $(\mathbb{Z}_n^*, \cdot)$

The set $\mathbb{Z}_n$ with the operation $\cdot$ (multiplication modulo n) is **not** a group:
For example, 0 has no multiplicative inverse b such that $b \cdot 0 \equiv 1$.

But the set $\mathbb{Z}_n^* := \{a \in \mathbb{Z}_n \mid \exists\, b \in \mathbb{Z}_n : \; b \cdot a = 1\}$ of invertible elements is a group.

Example $(\mathbb{Z}_{11}^*, \cdot)$:

$\cdot$	1	2	3	4	5	6	7	8	9	10
1	1	2	3	4	5	6	7	8	9	10
2	2	4	6	8	10	1	3	5	7	9
3	3	6	9	1	4	7	10	2	5	8
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
10	10	9	8	7	6	5	4	3	2	1

Recap: Invertible Elements modulo n and Euler phi-Function

- **Definition**: Integers a, b are co-prime if they have no common prime factor.

- **Theorem**: Element a has a multiplicative inverse mod n if a, n are co-prime. This inverse can be found with the Extended Euclidean Algorithm.

- **Definition**: The Euler phi-function $\varphi(n)$ is the number of integers in the range $1, \ldots, n-1$ which are co-prime to the integer n.
  - p prime: $\varphi(p) = p - 1$
  - $n = p \cdot q$ with p, q prime: $\varphi(n) = \varphi(p \cdot q) = (p-1) \cdot (q-1)$

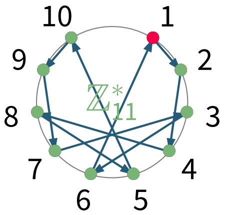
  Example: $\varphi(15) = (3-1) \cdot (5-1) = 8$: numbers $\{1, 2, 4, 7, 8, 11, 13, 14\}$

Recap: Generators and Euler's Theorem

- $\mathbb{Z}_n^*$ contains exactly the $\varphi(n)$ elements in $1, \ldots, n-1$ that are co-prime to n.
- **Euler's Theorem**: For all integers a and n that are co-prime: $a^{\varphi(n)} \equiv 1 \pmod{n}$
- **Definition**: If $\varphi(n)$ is the smallest integer $t > 1$ such that $a^t \equiv 1 \pmod{n}$, then a is called a generator of $\mathbb{Z}_n^*$.

Example: $a = 2$ is a generator of $\mathbb{Z}_{11}^*$, where $\varphi(11) = 10$:

- $2^1 = 2$
- $2^2 = 2 \cdot 2 = 4$
- $2^3 = 2 \cdot 4 = 8$
- $2^4 = 16 \equiv 5$
- $2^5 \equiv 10$
- $2^6 \equiv 20 \equiv 9$
- $2^7 \equiv 18 \equiv 7$
- $2^8 \equiv 14 \equiv 3$
- $2^9 \equiv 6$
- $2^{10} \equiv 12 \equiv 1$



The Discrete Logarithm Problem (DLP)

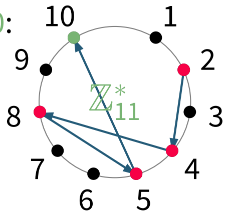
Discrete Logarithm Problem

Given a prime number p, a generator $g \in \mathbb{Z}_p^*$, and an element $y \in \mathbb{Z}_p^*$, find the integer $x \in \{0, \dots, p-2\}$ such that $\underbrace{g \cdot g \cdots g}_{x \text{ times}} = g^x \equiv y \pmod{p}$.

The DLP is believed to be hard in the group $(\mathbb{Z}_p^*, \cdot)$ for large primes p.

Example: Prime modulus $p = 11$, generator $g = 2$, and $y = 10$:

- $2^1 = 2$ ✗
- $2^2 = 4$ ✗
- $2^3 = 8$ ✗
- $2^4 = 16 \equiv 5 \pmod{11}$ ✗
- $2^5 = 32 \equiv 10 \pmod{11}$ ✓



The Integer Factorization Problem (IFP)

Integer Factorization Problem

Given $n \in \mathbb{N}$, find primes p_i and exponents $e_i \in \mathbb{N}$ such that $n = p_1^{e_1} \cdot p_2^{e_2} \cdot \dots \cdot p_k^{e_k}$

The IFP is believed to be hard if n is the product of two large primes: $n = p \cdot q$

Example: $n = 143 \quad \Rightarrow \quad n = p \cdot q = 11 \cdot 13$

Key Exchange

🔑

Establishing Secure Communication

Diffie–Hellman (DH) Key Exchange

- In 1976, Diffie and Hellman proposed the first asymmetric cryptosystem.
- DH and its relatives are the most relevant key-exchange algorithms in today's protocols. They allow Alice and Bob to derive a new shared secret key.
- Turing Award 2015
  Sometimes called Diffie–Hellman–Merkle (Merkle invented asymmetric crypto)

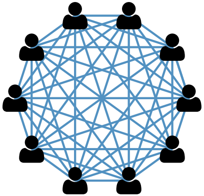
Whitfield Diffie

Martin Hellman

Ralph Merkle

Diffie–Hellman (DH) Key Exchange – Goal

- Adresses the key distribution problem



- If Alice and Bob want to start communicating, they exchange a few message to generate a shared secret key K to use for AEAD:

  Authentication ➜ Asymmetric crypto

  Key agreement ➜ Asymmetric crypto

  Actual communication ➜ Symmetric crypto

Diffie–Hellman (DH) Key Exchange – Definition

Diffie–Hellman Key Exchange

Choose a large prime p and a generator α of $\mathbb{Z}_p^*$ (public system parameters).

$a \in \{2, \ldots, p-2\}$

$$\xrightarrow{\alpha^a \pmod{p}}$$

$$\xleftarrow{\alpha^b \pmod{p}}$$

$b \in \{2, \ldots, p-2\}$

$K_{AB} \equiv (\alpha^b)^a \pmod{p}$

$K_{BA} \equiv (\alpha^a)^b \pmod{p}$

- Correctness: $K_{AB} \equiv (\alpha^b)^a \equiv (\alpha)^{b \cdot a} = (\alpha)^{a \cdot b} \equiv (\alpha^a)^b \equiv K_{BA}$, so both Alice and Bob derive the same key $K \equiv K_{AB} \equiv K_{BA}$
- We call a Alice's private key and α^a her public key (same for Bob's b and α^b)

Diffie–Hellman (DH) Key Exchange – Example

Diffie–Hellman Key Exchange

Choose a large prime $p = 11$ and a generator $\alpha = 2$ of $\mathbb{Z}_p^*$ (public parameters).

$$a = 3 \in \{2, \ldots, p-2\} \qquad \xrightarrow{\alpha^a \equiv 8}$$

$$\xleftarrow{\alpha^b \equiv 10} \qquad b = 5 \in \{2, \ldots, p-2\}$$

$$K_{AB} = (\alpha^b)^a \equiv 10^3 = 1000 \equiv 10$$

$$K_{BA} = (\alpha^a)^b \equiv 8^5 = 32768 \equiv 10$$

Diffie–Hellman (DH) Key Exchange – Security

Alice and Bob have no previous shared secrets. Eve knows all exchanged info:

- Parameters p and α
- Alice's public key $\alpha^a \pmod{p}$
- Bob's public key $\alpha^b \pmod{p}$

Eve would like to know the secret $K_{AB} \equiv (\alpha^a)^b \equiv (\alpha^a)^b \equiv \alpha^{a \cdot b}$.
This looks easy, but is generally believed to be a hard problem for large p.

Diffie–Hellman Problem (DHP)

Given generator $\alpha \in \mathbb{Z}_p^*$ and $\alpha^a \pmod{p}$, $\alpha^b \pmod{p}$, find $K_{AB} = \alpha^{a \cdot b}$.

Best known solution to DHP: find a from α^a, or b from α^b (= solve DLP in $\mathbb{Z}_p^*$).

Recommended key size: For 128-bit security, p should be about 3072 bits long.

Diffie–Hellman (DH) Key Exchange – Remarks

- The prime p and generator $\alpha \in \mathbb{Z}_p^*$ are public system parameters that can be the same for all users.
- Standards (NIST, ISO, ...) define parameters p, α for different security levels, how to encode values, how to use the resulting key K by hashing it to a suitable size, ...
- Modern protocols use ephemeral Diffie–Hellman (DHE) with temporary keypairs for forward secrecy.

Asymmetric Encryption

Confidentiality

Trapdoor One-way Functions

Asymmetric cryptography makes extensive use of "**one-way functions**":

easy to compute, hard to invert.

A "**trapdoor one-way function**" is a one-way function which can be inverted with an additional piece of information, the trapdoor information.

easy to lock

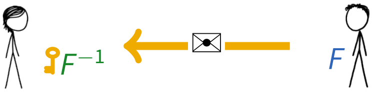
hard to open

unless you have the key

Asymmetric Encryption with Trapdoor One-Way Functions

- Receiver Alice creates and distributes a **trapdoor one-way function** F
- Sender Bob encrypt messages M by applying F (the public key):

$$C = F(M)$$



- Receiver Alice applies F^{-1} (the private key):

$$F^{-1}(C) = F^{-1}(F(M)) = M$$

- For Eve, it should be **computationally infeasible** to recover F^{-1} from F (or get M from C). However, everyone can compute a ciphertext C for any plaintext M.

Asymmetric Encryption – Algorithms and Keys

Key Generation

Alice generates a private key and corresponding public key. She distributes publicly and keeps safe.

Encrypt

With the public key, Bob (or anyone) encrypts a message M to a ciphertext C using $C = \mathcal{E}(M)$ and sends C to Alice.

Decrypt

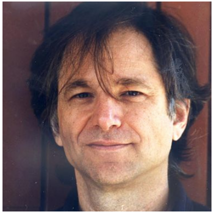
With her private key, Alice decrypts the ciphertext C to recover the message M using $\mathcal{D}(C) = M$

RSA (Rivest–Shamir–Adleman) Public-Key Encryption

- In 1977, Rivest, Shamir, and Adleman proposed one of the first public-key encryption schemes.
- RSA encryption as well as the related signature scheme are widely used.
- Turing Award 2002

Ron Rivest

Adi Shamir



Leonard Adleman

RSA Encryption (Rivest–Shamir–Adleman 1977)

Euler function:
$\varphi(pq) = (p-1)(q-1)$

Euler theorem:
if a, n are coprime, then
$a^{\varphi(n)} \equiv 1 \pmod{n}$

Key Generation

- Choose 2 large, random primes p, q
- Compute modulus $n = p \cdot q$
- Choose public exponent e co-prime to $\varphi(n)$
- Compute private exponent $d \equiv e^{-1} \pmod{\varphi(n)}$

public key $= (e, n)$ private key $= (d, n)$

Encrypt $\mathcal{E}(M)$

Encrypt message M:

$$C \equiv M^e \pmod{n}$$

Decrypt $\mathcal{D}(C)$

Decrypt ciphertext C:

$$M \equiv C^d \pmod{n} \equiv M^{e \cdot d} \equiv M^{1+k\varphi(n)} \equiv M$$

RSA Encryption – Example

Key Generation

- Choose 2 tiny, random primes $p = 3, q = 11$
- Compute modulus $n = p \cdot q = 33$
- Choose public exponent $e = 3$ co-prime to $\varphi(n) = (p-1)(q-1) = 2 \cdot 10 = 20$
- Compute private exponent $d \equiv e^{-1} \pmod{\varphi(n)} \equiv 7 \pmod{20}$ since $d \cdot e = 3 \cdot 7 = 21 = 20 + 1 \equiv 1 \pmod{20}$

Euler function:
$\varphi(pq) = (p-1)(q-1)$

Euler theorem:
if a, n are coprime, then
$a^{\varphi(n)} \equiv 1 \pmod{n}$

Encrypt $\mathcal{E}(M = 4)$

$$C \equiv M^e \pmod{n}$$
$$= 4^3 \equiv 31 \pmod{33}$$

Decrypt $\mathcal{D}(C = 31)$

$$M \equiv C^d \pmod{n} \equiv M^{e \cdot d} \equiv M^{1+k\varphi(n)} \equiv M$$
$$= 31^7 \equiv 4 \pmod{33}$$

RSA Encryption – Security

RSA Problem (RSAP)

Given modulus n, exponent e, ciphertext C: find M such that $M^e \equiv C \pmod{n}$.

- If we can solve factorization (IFP), we can recover p, q from n and break RSA
- The RSAP is believed to be as hard as the IFP and infeasible for large n.
- The modulus n must be large enough so that the runtime of the best factoring algorithms is not feasible for any attacker.

  Factoring record 2009: 768-bit modulus ($\approx$ 2000 CPU years)
- "Security level of k bits" = we estimate that factoring n takes more than 2^k time

Recommended key size: For 128-bit security, p should be about 3072 bits long.

RSA Encryption – Semantic Security

- There is a huge problem with this “textbook RSA”: It is **deterministic**.
- If the message has low entropy (e.g., $M \in \{\text{yes}, \text{no}, \text{maybe}\}$), the attacker can intercept C, **guess** M **and verify if** $C = \textbf{RSA}(M)$!
- We need a padding scheme to make RSA “**semantically secure**”:

Indistinguishability (under Adaptive Chosen-Ciphertext Attack)

An attacker who knows the public key, chooses 2 messages M_0, M_1, and gets ciphertext C **can not distinguish** $C = E(M_0)$ or $C = E(M_1)$, even if they can ask for decryption of any $C \neq C^*$.

RSA Encryption – Padding for Semantic Security

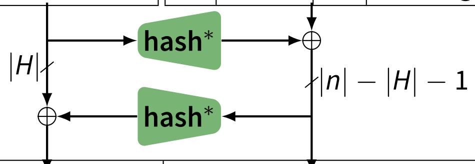
PKCS #1 (**P**ublic-**K**ey **C**ryptography **S**tandard) defines 2 **RSA E**ncryption **S**chemes (**RSAES**):

- **RSAES-PKCS1-v1_5** (deprecated):

  $$C = \textbf{RSA}\Big(\boxed{\texttt{00 02}}\boxed{\geq 8\ \texttt{random bytes}}\boxed{\texttt{00}}\boxed{\texttt{message}\ M}\Big)$$

- **RSAES-OAEP** ("optimal asymmetric encryption padding"):

  1. Compute $H = \textbf{hash}(\texttt{label}\ L)$. Generate `random seed` ($|H|$ bytes).
  2. Compute $\boxed{\texttt{random seed}}\ \boxed{H}\boxed{0\cdots0}\boxed{01}\boxed{\texttt{message}\ M}$

  3. $C = \textbf{RSA}\Big(\boxed{\texttt{00}}\boxed{\texttt{masked seed}}\boxed{\texttt{masked data}}\Big)$

Signatures

Authenticity

Signatures – Algorithms in a Signature Scheme

Key Generation

Alice generates a private key and corresponding public key .
She distributes publicly and keeps safe.

Sign

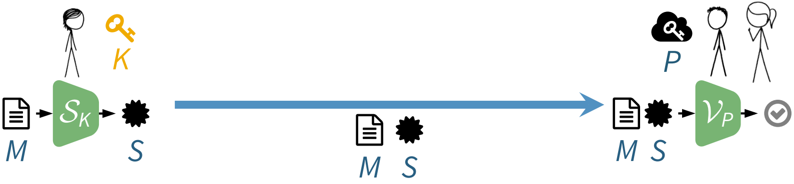
With her private key , Alice computes the signature $\mathcal{S}(M) = S$ ✸
of a message M. She transmits , ✸ to the recipient(s).

Verify

With the public key , Bob (or anyone) can verify the signature:
$\mathcal{V}(M, T) \in \{✔, ✖\}$

Signatures – Definition and Application

Signatures: private key K and public key P



Digital signatures ensure

- Sender authentication
- Message integrity
- Non-repudiation

Signatures – Security

- It must be easy to compute S using the private key
- It must be easy to verify S using the public key
- It must be hard to compute S without the private key (forgery)
  even if the attacker chooses the message and knows previous signatures

This is achieved using complexity-theoretically hard problems such as

- IFP: Integer factorization problem
- DLP: Discrete logarithm problem

RSA Signatures (Rivest–Shamir–Adleman 1977)

Euler function:
$\varphi(pq) = (p-1)(q-1)$

Euler theorem:
if a, n are coprime, then
$a^{\varphi(n)} \equiv 1 \pmod{n}$

Key Generation

- Choose 2 large, random primes p, q
- Compute modulus $n = p \cdot q$
- Choose public exponent e co-prime to $\varphi(n)$
- Compute private exponent $d \equiv e^{-1} \pmod{\varphi(n)}$

public key $= (e, n)$ private key $= (d, n)$

Sign $\mathcal{S}(M)$

Compute signature S:

$$S \equiv M^d \pmod{n}$$

Verify $\mathcal{V}(M, S)$

Verify that

$$M \stackrel{?}{\equiv} S^e \pmod{n} \equiv M^{d \cdot e} \equiv M^{1+k\varphi(n)} \equiv M$$

RSA Signatures – Security

⚠ The message M is recoverable from the signature S as $M \equiv S^e$ ⚠
$\rightarrow$ An attacker can easily generate valid pairs (M, S)!

Solution: Sign the hash of the message ("signature with appendix")

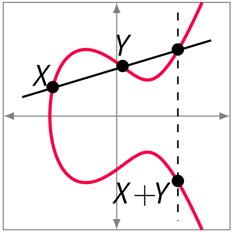
PKCS #1 defines 2 **RSA S**ignature **S**chemes with **A**ppendix (**RSASSA**):

- **RSASSA-PKCS1-v1_5** (legacy):
  1. Compute **hash**(M)
  2. $S =$ **RSA-Sign**$\left(\boxed{\texttt{00 01} \mid \texttt{FF}\cdots\texttt{F} \mid \texttt{00} \mid \textbf{hash}(M)} \right)$
- **RSASSA-PSS** (provably secure "**p**robabilistic **s**ignature **s**cheme")

Discussion

ECC = Elliptic Curve Cryptography

- $\mathbb{Z}_p^*$ is not the only useful group for the Discrete Logarithm Problem.
- An attractive alternative is the Elliptic Curve group, where each element is not an integer but a 2-dimensional point with two integer coordinates. The group operation is addition with special point addition formulas.



EC Discrete Logarithm Problem (ECDLP)

Given points P, Q on an elliptic curve with

$$Q = k \cdot P = \underbrace{P + P + \ldots + P}_{k \text{ times}},$$

Find k.

Today’s and Tomorrow’s Public-Key Schemes – Security

(EC)DSA is a signature based on the (Elliptic-Curve) Discrete Logarithm Problem.

Estimated security levels for different modulus bitsizes (NIST SP 800-57):

Security level	RSA	DH, DSA	ECDH, ECDSA
80 bits	1024	1024	160
112 bits	2048	2048	224
128 bits	**3072**	**3072**	**256**
192 bits	7680	7680	384
256 bits	15360	15360	512

- A fast quantum computer could solve these hard problems much faster
- Ongoing research in efficient post-quantum secure signature schemes

Practical considerations: What you should be aware of

1. Signatures do not protect confidentiality.
   - ➔ If necessary, combine with encryption
2. Signatures are only as authentic as the public keys.
   - ! A secure public-key infrastructure (PKI) is essential
   - ➔ Certification Authority (CA) or Web-of-Trust
3. Signatures just sign a bitstring, not its “meaning” or context.
   - ! Metadata like time (old code “updates”), program version (≠), file format (polyglots =), . . .

. . .

Conclusion

Conclusion

- Establishing a secure communication channel
  - Authentication ➔ Asymmetric crypto
  - Key agreement ➔ Asymmetric crypto
  - Actual communication ➔ Symmetric crypto
- Important asymmetric schemes (key sizes: 3072+ bits)
  - Diffie–Hellman (DH) key exchange
  - RSA encryption
  - RSA signatures

Questions