IDATT2503, Part 2 Cryptography

Lecture 1: Introduction, information. Ideas and concepts examplified through some classical ciphers.

Hans J. Rivertz

10th. October

Information

- The lectures will be in the weeks 41–46, repetition in week 47.
- 6 assignments, all must be done, 4 must be approved.
- Learning resources: The lecture notes, assignments, and some additional material. Suggestions for literature will be given in the slides.

Assignments

Hand in the files with your answers in Blackboard.

Please do your own work!

Exam

4 hours, 50% cryptography

Bring no written written material to the exam. However, a cheat sheet will be attached in Inspera. It covers what I consider not so important to memorize. The focus will be on testing your understanding. It contains

- Number theory,
- Algorithms
- Some definitions

Teaching material and resources

The lecture notes and the assignments defines the curriculum, with some additions. There are many freely available online sources.

- https://www.crypto101.io/
- https://joyofcryptography.com/
- Introduction to modern Cryptography https://web.cs.ucdavis.edu/ rogaway/classes/227/spring05/book/main.pdf

Software

- **CrypTool-Online** Apps to explore, play around with, and learn about cryptology. For students, teachers, and anyone interested. https://www.cryptool.org/en/cto/
- http://practicalcryptography.com/

Book: Cryptography and network security, by W. Stallings

Other resources might be given later.

Today's lecture

- Historical ciphers and why they do not provide security
- Concepts and features of old ciphers that modern ciphers have kept.
- The mathematics behind all modern cryptography
- Some statistics and probability theory for cryptanalysis

Teaching goal

You will learn about

- A mathematical foundation
  - Number theory
  - Combinatorics and simple probability theory to understand security of cryptography systems.
- Concepts in modern cryptography
- Some old and modern cryptosystems
  - Symmetric ciphers
  - Asymmetric ciphers, (public key)
- Define and understand what security means in a cryptography setting
- Cryptanalysis
  - Different attack models and practical example

Introduction

1 Introduction

2 Some historical ciphers

3 Other important points

Some necessities

Modern cryptography relies heavily on mathematics. Key areas to be familiar with are:

- **Number representations** (binary, hexadecimal – assumed known).
- **Number theory:** modular arithmetic, primes, and key results (covered in course).
- **Probability & statistics:** useful for defining security and cryptanalysis.
- **Programming:** helpful for understanding algorithms (used in assignments, not exams). Any common language is fine.

What is Cryptography

Cryptography (Greek: Κρυπτογραφία "hidden writing") is the science of securing information against unauthorized access or tampering. It ensures:

- **Secrecy** (confidentiality)
- **Integrity** (no tampering)
- **Authenticity** (proof of sender)
- **Non-repudiation** (cannot deny sending)

Cryptanalysis is the reverse: **breaking ciphers**, extracting or altering data, or impersonating others. **Cryptology** covers both cryptography and cryptanalysis.

[0] "**crypto**" often refers to **cryptocurrency**.

Information security

Information Security aims to reduce vulnerabilities in information assets.

- **Vulnerability:** exploitable weakness.
- **Assets:** data, software, hardware, people, buildings.
- **Threat:** potential security violation by an adversary.

Cryptography is one tool within information security.

Recommended video: First part of https://youtu.be/o1x_Oa0XiDI?si=Icky38jfIYIXdCpf

From Historical to Modern Cryptography

- Early cryptography relied on tricks and belief (e.g., *Vigenère cipher*, thought unbreakable until 1863).
- Modern cryptography defines **security precisely** through:
  - Clear assumptions, mathematical proofs.
  - Modeling attackers capabilities and resources and statistics.
  - Reliance on open problems (e.g., $\mathbf{P} \neq \mathbf{NP}$).

Modern Cryptography

Modern Cryptography is a scientific discipline characterized by:

- Rigorous analysis with a solid theoretical foundation.
- Well-defined attack models and provable security under given assumptions.
- Beyond secrecy, it also ensures:
  - Data integrity
  - Digital signatures
  - Non-repudiation
  - And more...
- Focused on the design, analysis, and implementation of mathematical and technical methods to secure information, systems, and distributed computations against attacks.

Cryptography is widespread

- Secure transactions over open networks
- Encryption of stored information (e.g. disk encryption)
- Digitally signed software updates
- Password management
- Cryptocurrency (not syllabus)

School versions vs. real-world use

- Textbook examples may differ from real cryptographic protocols.
- We will focus on the key building blocks (cryptographic primitives).
- These form protocols that secure whole processes (e.g., SSL).
- A system's security relies on certain assumptions holding true. Incorrect use of otherwise secure mechanisms can weaken or break security.

Cipher classification

1. **Symmetric** (shared key):
   - Secrecy → encryption/decryption
   - Integrity → Message authentication codes (MACs)

   AES
2. **Asymmetric** (public/private key):
   - Secrecy → RSA, ElGamal
   - Integrity → digital signatures
3. **Important key ingredients**:
   - random numbers
   - random functions
   - hash functions

Character roles in cryptography

In crypto analysis, Alice and Bob are the heroes, Eve is the eavesdropper, Mallory the villain.

- **Alice & Bob:** Parties communicating securely
- **Eve:** Eavesdropper (usually just listens, sometimes more)
- **Mallory:** Active attacker who tampers, forges, or injects messages — more dangerous

Alice

EVE

Bob

https://en.wikipedia.org/wiki/Alice_and_Bob

Some historical ciphers

1 Introduction
2 Some historical ciphers
3 Other important points

Why study historical ciphers?

- They aren't secure—but are useful for introducing important concepts.
- Show that designing secure ciphers is hard.
- Hopefully they're fun to play with!

Caesar Cipher: Encryption

The alphabet has a fixed order:

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

To encrypt, replace each letter with the one **3 positions later** in the alphabet. Wrap around to the start if needed.

Example: Encrypt the word “ENCRYPT” using this rule.

E N C R Y P T

Caesar Cipher: Encryption

The alphabet has a fixed order:

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

To encrypt, replace each letter with the one **3 positions later** in the alphabet. Wrap around to the start if needed.

Example: Encrypt the word "ENCRYPT" using this rule.

E N C R Y P T

H

Caesar Cipher: Encryption

The alphabet has a fixed order:

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

To encrypt, replace each letter with the one **3 positions later** in the alphabet. Wrap around to the start if needed.

Example: Encrypt the word "ENCRYPT" using this rule.

E N C R Y P T

H Q

Caesar Cipher: Encryption

The alphabet has a fixed order:

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

To encrypt, replace each letter with the one **3 positions later** in the alphabet. Wrap around to the start if needed.

Example: Encrypt the word "ENCRYPT" using this rule.

E N C R Y P T

H Q F

Caesar Cipher: Encryption

The alphabet has a fixed order:

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

To encrypt, replace each letter with the one **3 positions later** in the alphabet. Wrap around to the start if needed.

Example: Encrypt the word "ENCRYPT" using this rule.

E N C R Y P T

H Q F U

Caesar Cipher: Encryption

The alphabet has a fixed order:

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

To encrypt, replace each letter with the one **3 positions later** in the alphabet. Wrap around to the start if needed.

Example: Encrypt the word "ENCRYPT" using this rule.

E N C R Y P T

H Q F U B

Caesar Cipher: Encryption

The alphabet has a fixed order:

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

To encrypt, replace each letter with the one **3 positions later** in the alphabet. Wrap around to the start if needed.

Example: Encrypt the word "ENCRYPT" using this rule.

E N C R Y P T

H Q F U B S

Caesar Cipher: Encryption

The alphabet has a fixed order:

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

To encrypt, replace each letter with the one **3 positions later** in the alphabet. Wrap around to the start if needed.

Example: Encrypt the word "ENCRYPT" using this rule.

E N C R Y P T

H Q F U B S W

Caesar Cipher: Decryption

The alphabet has a fixed order:

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

To encrypt, replace each letter with the one **3 positions before** in the alphabet. Wrap around to the end if needed.

Example: Decrypt the code "FOHRSDWUD" using this rule.

F O H R S D W U D

Caesar Cipher: Decryption

The alphabet has a fixed order:

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

To encrypt, replace each letter with the one **3 positions before** in the alphabet. Wrap around to the end if needed.

Example: Decrypt the code “FOHRSDWUD” using this rule.

F O H R S D W U D

$\downarrow$

C

Caesar Cipher: Decryption

The alphabet has a fixed order:

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

To encrypt, replace each letter with the one **3 positions before** in the alphabet. Wrap around to the end if needed.

Example: Decrypt the code "FOHRSDWUD" using this rule.

F O H R S D W U D

C L

Caesar Cipher: Decryption

The alphabet has a fixed order:

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

To encrypt, replace each letter with the one **3 positions before** in the alphabet. Wrap around to the end if needed.

Example: Decrypt the code "FOHRSDWUD" using this rule.

F O H R S D W U D

C L E

Caesar Cipher: Decryption

The alphabet has a fixed order:

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

To encrypt, replace each letter with the one **3 positions before** in the alphabet. Wrap around to the end if needed.

Example: Decrypt the code "FOHRSDWUD" using this rule.

F O H R S D W U D

C L E O

Caesar Cipher: Decryption

The alphabet has a fixed order:

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

To encrypt, replace each letter with the one **3 positions before** in the alphabet. Wrap around to the end if needed.

Example: Decrypt the code "FOHRSDWUD" using this rule.

F O H R S D W U D

C L E O P

Caesar Cipher: Decryption

The alphabet has a fixed order:

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

To encrypt, replace each letter with the one **3 positions before** in the alphabet. Wrap around to the end if needed.

Example: Decrypt the code “FOHRSDWUD” using this rule.

F O H R S D W U D

C L E O P A

Caesar Cipher: Decryption

The alphabet has a fixed order:

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

To encrypt, replace each letter with the one **3 positions before** in the alphabet. Wrap around to the end if needed.

Example: Decrypt the code “FOHRSDWUD” using this rule.

F O H R S D W U D

C L E O P A T

Caesar Cipher: Decryption

The alphabet has a fixed order:

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

To encrypt, replace each letter with the one **3 positions before** in the alphabet. Wrap around to the end if needed.

Example: Decrypt the code "FOHRSDWUD" using this rule.

F O H R S D W U D

C L E O P A T R

Caesar Cipher: Decryption

The alphabet has a fixed order:

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

To encrypt, replace each letter with the one **3 positions before** in the alphabet. Wrap around to the end if needed.

Example: Decrypt the code "FOHRSDWUD" using this rule.

F O H R S D W U D

C L E O P A T R A

Shift or rotation cipher

- **Caesar cipher:** shifts letters by 3. The security relies only in a secret algorithm.
- **Shift cipher:** shifts letters by a **random chosen** secret number k (the key). *Example:* with k = 5,
  $A \to F, B \to G, C \to H, \cdots, Z \to E$
- **Kerckhoff's principle:** security should come only from keeping a **random chosen** key secret, not from hiding how the cipher works.
- The shift cipher fits Kerckhoff's principle, but **is it secure?**

Shift or rotation cipher

- **Caesar cipher:** shifts letters by 3. The security relies only in a secret algorithm.
- **Shift cipher:** shifts letters by a **random chosen** secret number k (the key). *Example:* with k = 5, $A \to F, B \to G, C \to H, \cdots, Z \to E$
- **Kerckhoff's principle:** security should come only from keeping a **random chosen** key secret, not from hiding how the cipher works.
- The shift cipher fits Kerckhoff's principle, but **is it secure?** **No! Only 25 keys to try**

Definition of Cipher

To formalize what we have so far, a cipher consists of:

- An **alphabet** (set of symbols) Σ.
- A **set of plaintexts** $\mathcal{M}$ that can be encrypted (often all strings over Σ, i.e., Σ^*).
- A **set of ciphertexts** $\mathcal{C}$.
- A **set of keys** $\mathcal{K}$.
- A (pseudo-)random **key generator** $K \leftarrow Gen$.
- Two algorithms:
  - **Encryption:** $\mathcal{E} : \mathcal{K} \times \mathcal{M} \to \mathcal{C}$
  - **Decryption:** $\mathcal{D} : \mathcal{K} \times \mathcal{C} \to \mathcal{M}$

Requirement: For each key $k \in \mathcal{K}$, decryption must be the inverse of encryption:

$$\mathcal{D}_k(\mathcal{E}_k(m)) = m \quad \text{for all } m \in \mathcal{M}.$$

$\mathbb{Z}/n\mathbb{Z}$: Modular Arithmetic

- In modular arithmetic, numbers are considered **equivalent** if their difference is divisible by n. equivalent modulo n
- Example (mod 5):

$$2,\ 7,\ 12,\ \ldots,\ 2+5k \quad (k \in \mathbb{Z})$$

are all equivalent (**in the same equivalence class**).

- We write: $\{\ldots, -3, 2, 7, 12, 17, \ldots\}$

$$a \equiv b \pmod{n} \quad \Leftrightarrow \quad n \mid (a-b).$$

- The notation a mod n or $a\%n$ means the **remainder** of a when divided by n, always in the range 0 to $n-1$.

Quotient-Remainder Theorem

For any positive integer n and any integer a, there exist **unique integers** k and r such that

$$a = kn + r \quad \text{with} \quad 0 \leq r < n.$$

Here, r is the **remainder** when a is divided by n, often written as

$$a \bmod n = r \quad \text{or, in programming,} \quad \texttt{a\%n} = \texttt{r}.$$

More Modular Arithmetic

Modulo Calculus: Reduce numbers to their **residues modulo** n at each step.

Most arithmetic rules still hold modulo n.

Examples:

- $(7 \cdot 4) \cdot 3 \equiv 8 \cdot 3 \equiv 4 \pmod{10}$
  $7 \cdot (4 \cdot 3) \equiv 7 \cdot 2 \equiv 4 \pmod{10}$
- $3 \cdot (7+9) \equiv 3 \cdot 6 = 8 \pmod{10}$
  $3 \cdot (7+9) \equiv 1 + 7 \equiv 8 \pmod{10}$

Powers:

$$11^{100} \bmod 10 = 1, \quad 10^{100} \bmod 10 = 0$$

Note: Exponents **cannot** be reduced modulo n, e.g.,

$$2^8 \bmod 7 \neq 2^1 \bmod 7$$

Formalizing the Rotation Cipher

For a single character (letter):

$$\Sigma = \mathcal{K} = \mathcal{M} = \mathcal{C} = \mathbb{Z}_{26}$$

Encryption and decryption functions:

$$\mathcal{E} : \mathcal{K} \times \mathcal{M} \to \mathcal{C}, \quad \mathcal{D} : \mathcal{K} \times \mathcal{C} \to \mathcal{M}$$

$$\mathcal{E}_K(M) = (M + K) \bmod 26, \quad \mathcal{D}_K(C) = (C - K) \bmod 26$$

For sequences (strings) of arbitrary length:

$$\mathcal{M} = \mathcal{C} = \Sigma^*$$

Apply the algorithm **character by character**:

$$m = m_1 m_2 \ldots m_l \implies \mathcal{E}_K(m) = \mathcal{E}_K(m_1)\, \mathcal{E}_K(m_2) \ldots \mathcal{E}_K(m_l)$$

Cryptanalysis of ROT

Attack model: Cipher-text only (COA) — attacker sees only ciphertexts.

Brute force: Small keyspace $\rightarrow$ try every key.

Assumption: Attacker can recognize correct plaintext (reasonable for natural language; for other data, unencrypted outputs usually show identifiable structure).

ROT and Perfect Secrecy

ROT gives **perfect secrecy** if a *single character* is encrypted with a uniformly random key. In this case, the ciphertext reveals no extra information about the plaintext.

Problem 1: For perfect secrecy, the one-time pad key must be as long as the message and exchanged securely.

Problem 2: Reusing the same key for multiple messages breaks secrecy.

USSR Key Reuse: The USSR compromised one-time pad security by reusing keys, as revealed in the Venona project.

Increasing the Number of Keys

ROT has too few keys. Possible extensions:

1. **Larger function families**
   - Affine cipher: $E_k(x) = ax + b \pmod{n}$, with key $k = (a, b)$
   - Other formulas with more parameters
   - Ultimately: arbitrary random permutation of the alphabet
2. **Block ciphers**
   - Encrypt multiple characters at once
   - Defined by a formula or algorithm
   - Ultimately: random permutation of all possible blocks
3. **Stream ciphers**
   - Key varies with position in the stream
   - Typically simple operations (e.g., XOR per character)

Simple (Mono-alphabetical) Substitution Ciphers

We can greatly increase the number of keys by allowing **arbitrary permutations of the alphabet**:

- Alphabet: Σ
- Messages and ciphertexts: $\mathcal{M} = \mathcal{C} = \Sigma^*$ (all sequences)
- Key space: $\mathcal{K} =$ all permutations of Σ
- Key generation: $\pi \leftarrow$ Gen $p \in \mathcal{K}$ (random permutation)

Example:

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
H	L	B	W	I	G	N	T	Q	A	R	Y	Z	D	M	P	E	J	V	O	F	X	S	K	C	U

Cryptanalysis of Simple Substitution Cipher

For a 26-letter alphabet, the number of keys is

$$26! = 26 \cdot 25 \cdot \cdots \cdot 2 \cdot 1 \approx 4 \cdot 10^{26} \text{ (about 89 bits)}$$

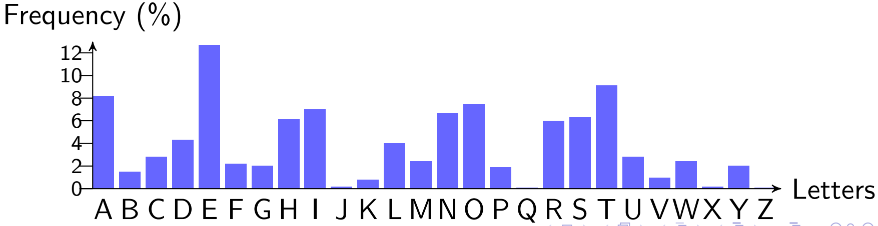
Brute-force security: Practically impossible to try all keys; on average, half would need to be tested.

But: Large keyspace does not guarantee safety against other cryptanalysis methods.

Cryptanalysis Using Frequency Analysis

Languages exhibit statistical regularities. Simple substitution ciphers preserve **relative letter frequencies**.

- Frequency analysis of single letters or combinations helps reverse the permutation.
- Encrypted spaces are usually the most frequent character and can aid analysis of words.
- Effectiveness depends on text length and type.
- Since only the ordering changes, comparing frequencies often reveals the substitution.



Frequency Analysis

See detailed explanations at:

- https://crypto.interactive-maths.com/frequency-analysis-breaking-the-code.html
- http://practicalcryptography.com/cryptanalysis/stochastic-searching/cryptanalysis-simple-substitution-cipher/

- Compare ciphertext letter frequencies to typical language frequencies to make an initial guess.
- Examining letter combinations can refine the substitution.
- Iteratively modify the substitution until the text becomes meaningful.
- Effective only for sufficiently long ciphertexts (typically ≥ 50 characters).

Attack models

- **COA** Ciphertext-only: attacker sees only ciphertexts.
- **KPA** Known-plaintext: attacker knows one or more plaintext–ciphertext pairs.
- **CPA** Chosen-plaintext: attacker can obtain ciphertexts for plaintexts she chooses (CPA2 $=$ adaptive).
- **CCA** Chosen-ciphertext: attacker can obtain plaintexts for ciphertexts she submits.

Note: the amount of available text also affects attack feasibility.

Other attack models

Side-channel attacks: e.g. timing, power, electromagnetic or fault attacks. They exploit implementation leaks (hardware/software) rather than weaknesses in the cipher itself, so they are not covered here.

Adversary Goals

A successful attack may aim to:

- Recover all or part of a plaintext.
- Recover all or part of a key (to decrypt or forge messages).
- Modify a message undetectably (integrity/forgery), possibly without decrypting it.

Note: Attack capabilities (e.g. degree of control over modifications) and non-secrecy targets vary by application.

Cryptanalysis: Simple Substitution

- **Known-plaintext:** Reveals mappings for observed letters — helps recover the rest.
- **Chosen-plaintext:** Can reveal the entire key (by supplying all alphabet letters).
- **Chosen-ciphertext:** Decryption oracle exposes full key (by decrypting a ciphertext alphabet).

Varieties of Substitution Ciphers

- **Simple (monoalphabetic):** Fixed substitution for each letter.
- **Polyalphabetic (e.g., Vigenère):** Uses different substitutions depending on position.
- **Polygraphic:** Substitutes groups of letters (e.g., digraphs like aa, ab, . . .), increasing key space. Frequency analysis still works but requires more ciphertext.

Block Ciphers

- Messages are divided into fixed-length blocks, similar to substitution ciphers.
- The last block may require padding to reach full length.
- Key space grows with block size — consider blocks in **bits** to evaluate the number of possible keys.

Number of Possible Keys in a Block Cipher

- **Block length (bits)** n
- **Number of permutations** $2^n!$
- **Bits to represent permutation** $\log_2(2^n!)$

n	$2^n!$	$\log_2(2^n!)$
5	$2.6 \cdot 10^{35}$	113
8	$8.5 \cdot 10^{506}$	1678
16		$9.5 \cdot 10^5$
256		$3 \cdot 10^{79}$

- Longer blocks improve security by making statistical analysis harder.
- Arbitrary permutations for large blocks are impractical to store.
- Solution: generate pseudorandom permutations from a smaller key — topic for next week.

Other important points

1 Introduction

2 Some historical ciphers

3 Other important points

Practical Considerations for $\mathcal{E}, \mathcal{D}$, and Key Generation

- Encryption $\mathcal{E}$ and decryption $\mathcal{D}$ must be efficiently computable.
- Key generator Gen must produce truly random keys; predictable keys weaken security.
- See reference video for examples (timestamp needed).

Injectivity of Encryption

- Encryption must be **injective** so decryption is well-defined; often it is **bijective**.
- **Homophonic ciphers** map a plaintext character to multiple ciphertext symbols, making encryption nondeterministic while keeping decryption possible.
- Homophonic ciphers are useful to flatten letter frequencies; not covered in this course.

Kerckhoffs's Principle

Security should rely only on keeping the **key** secret; the algorithm is assumed public.

Some advantages are:

- Easier to update keys than redesign algorithms if compromised.
- Keys are simpler to keep secret than entire algorithms.
- Public algorithms can be thoroughly tested and optimized.
- Facilitates standardization with known, reliable characteristics.