



Applied!

Data & Network Security

Behnam Amiri

ans.dailysec.ir

aNetSec.github.io

Spring 2025

Digital Signatures

Why digital signature?

- Certain that the message is indeed from him/her.
 - Bob uses hash function, to generate a hash value for the message.
 - Encrypt hash with his private key.
- Can't make fake message and assign to sender.
- Sender can't deny send message!

Digital Signature

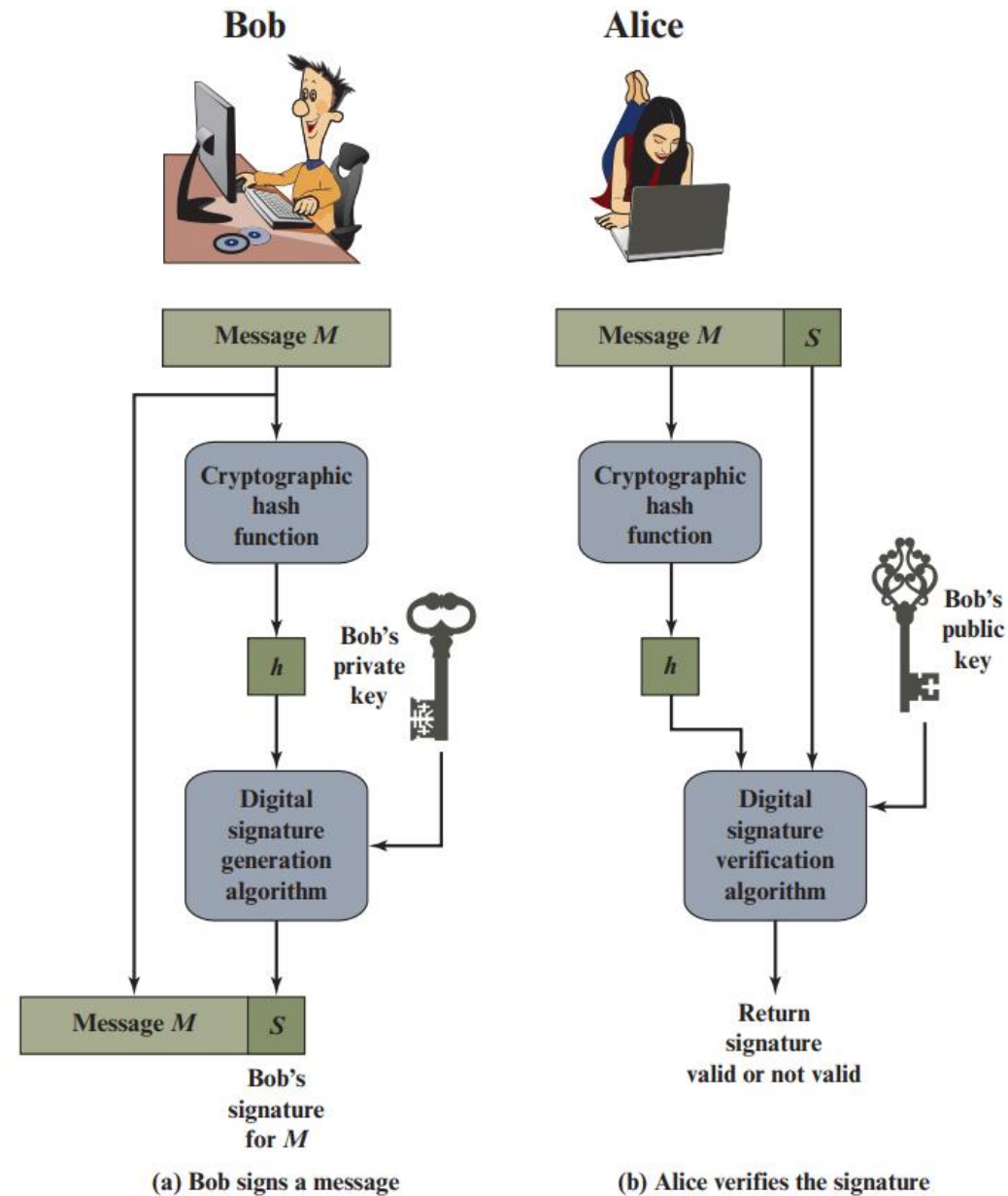


Figure 13.1 Simplified Depiction of Essential Elements of Digital Signature Process

Digital Signature properties

- It must verify the author and the date and time of the signature.
- It must authenticate the contents at the time of the signature.
- It must be verifiable by third parties, to resolve disputes.

Digital Signature Requirements

- The signature must be a bit pattern that depends on the message being signed.
- The signature must use some information only known to the sender to prevent both forgery and denial.
- It must be relatively easy to produce the digital signature.
- It must be relatively easy to recognize and verify the digital signature.
- It must be computationally infeasible to forge a digital signature, either by constructing a new message for an existing digital signature or by constructing a fraudulent digital signature for a given message.
- It must be practical to retain a copy of the digital signature in storage.

Digital Signature properties

- It must verify the author and the date and time of the signature.
- It must authenticate the contents at the time of the signature.
- It must be verifiable by third parties, to resolve disputes.

Digital Signature usage

- Sign iso images
- Sign packages
- Sign updates

← → ↻ <https://cdimage.debian.org/debian-cd/current/amd64/iso-cd/>

Only the first few images are available! Where are the rest?

We don't store/serve the full set of ISO images for all architectures, to reduce the amount of space taken

Non-free Firmware

This Debian image build only includes Free Software where possible. However, many systems include those firmware files for those cases. See the Debian Wiki [non-free firmware](#) page for more information

Other questions?

See the Debian CD [FAQ](#) for lots more information about Debian CDs and installation.

The images here were put together by the [Debian CD team](#), using debian-cd and other software.

[Name](#)


⏪ [Parent Directory](#)

 [SHA256SUMS](#)

 [SHA256SUMS.sign](#)

 [SHA512SUMS](#)

 [SHA512SUMS.sign](#)

 [debian-12.9.0-amd64-netinst.iso](#)

 [debian-edu-12.9.0-amd64-netinst.iso](#)

 [debian-mac-12.9.0-amd64-netinst.iso](#)

The DSA standard

- NIST has published Federal Information Processing Standard FIPS 186, known as the Digital Signature Algorithm (DSA).
- The DSA makes use of the Secure Hash Algorithm (SHA).

Compare

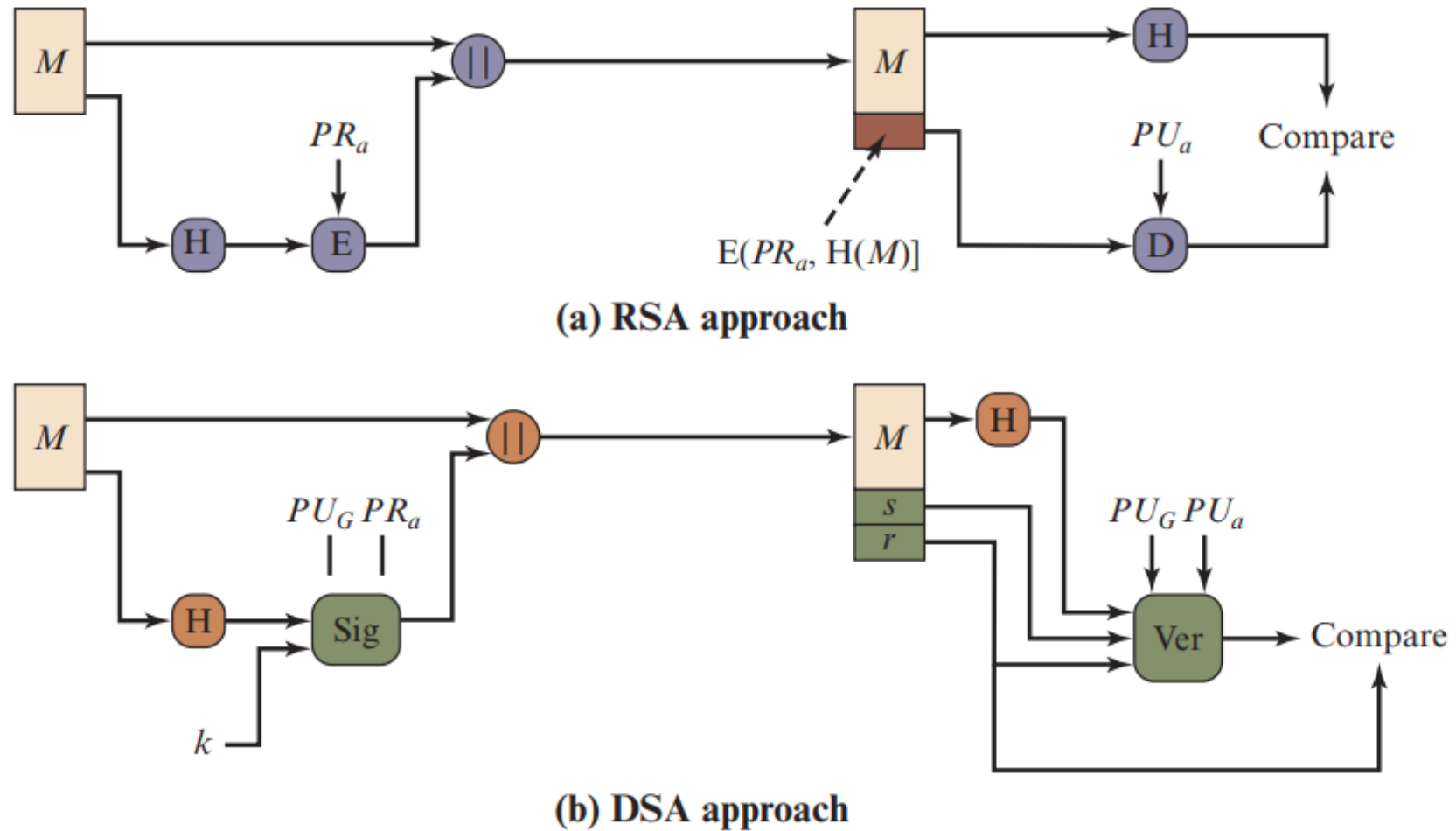


Figure 13.2 Two Approaches to Digital Signatures

DSA Components

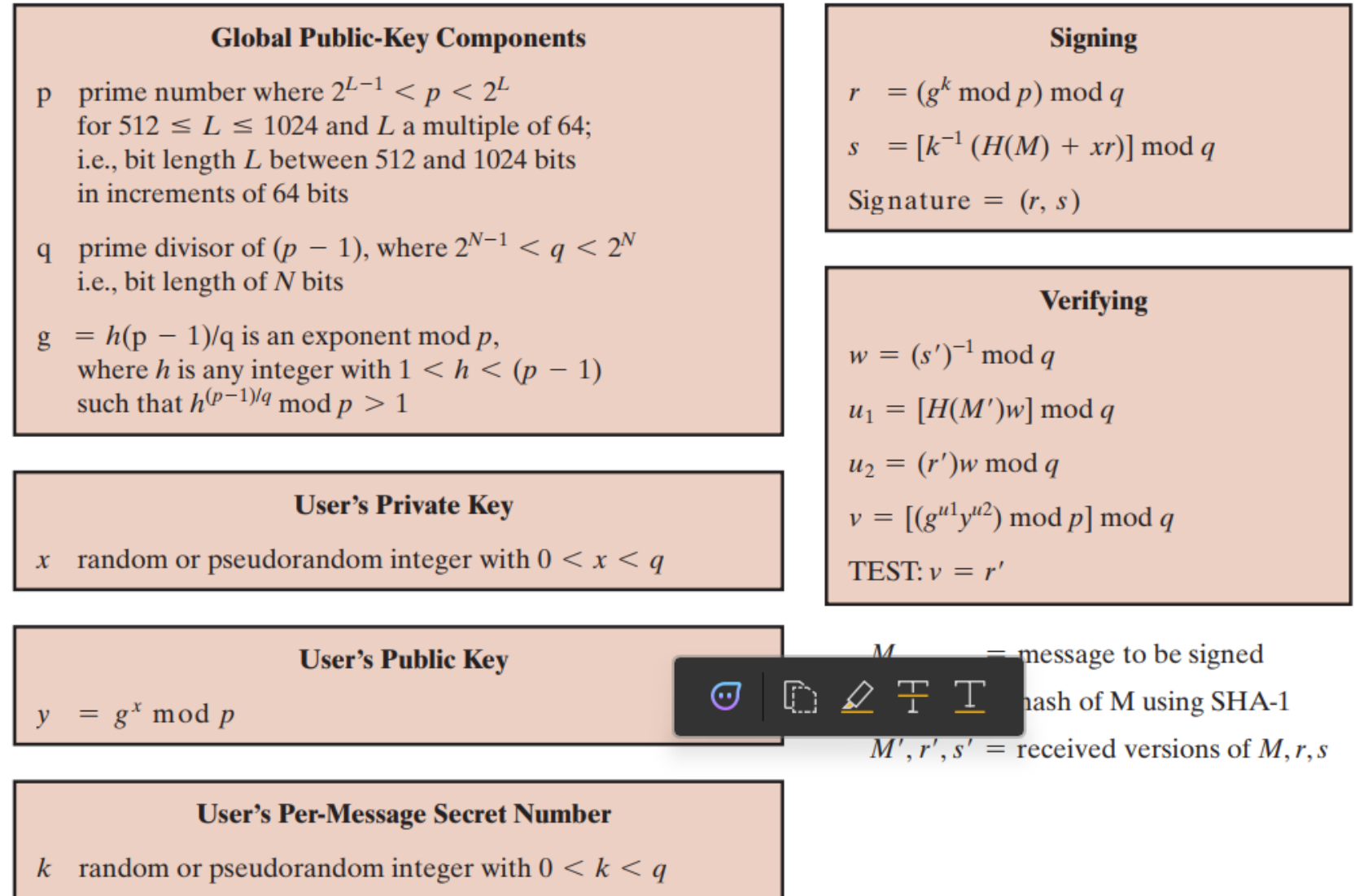


Figure 13.3 The Digital Signature Algorithm (DSA)

Elliptic Curve Digital Signature Algorithm

- The 2009 version of FIPS 186 includes a new digital signature technique based on [elliptic curve cryptography](#), known as the Elliptic Curve Digital Signature Algorithm (ECDSA).

Overview of the process in ECDSA

1. All those participating in the digital signature scheme use the same global domain parameters, which define an elliptic curve and a point of origin on the curve.
2. A signer must first generate a public, private key pair. For the private key, the signer selects a random or pseudorandom number. Using that random number and the point of origin, the signer computes another point on the elliptic curve. This is the signer's public key.
3. A hash value is generated for the message to be signed. Using the private key, the domain parameters, and the hash value, a signature is generated. The signature consists of two integers, r and s .
4. To verify the signature, the verifier uses as input the signer's public key, the domain parameters, and the integer s . The output is a value v that is compared to r . The signature is verified if $v = r$.

ECDSA

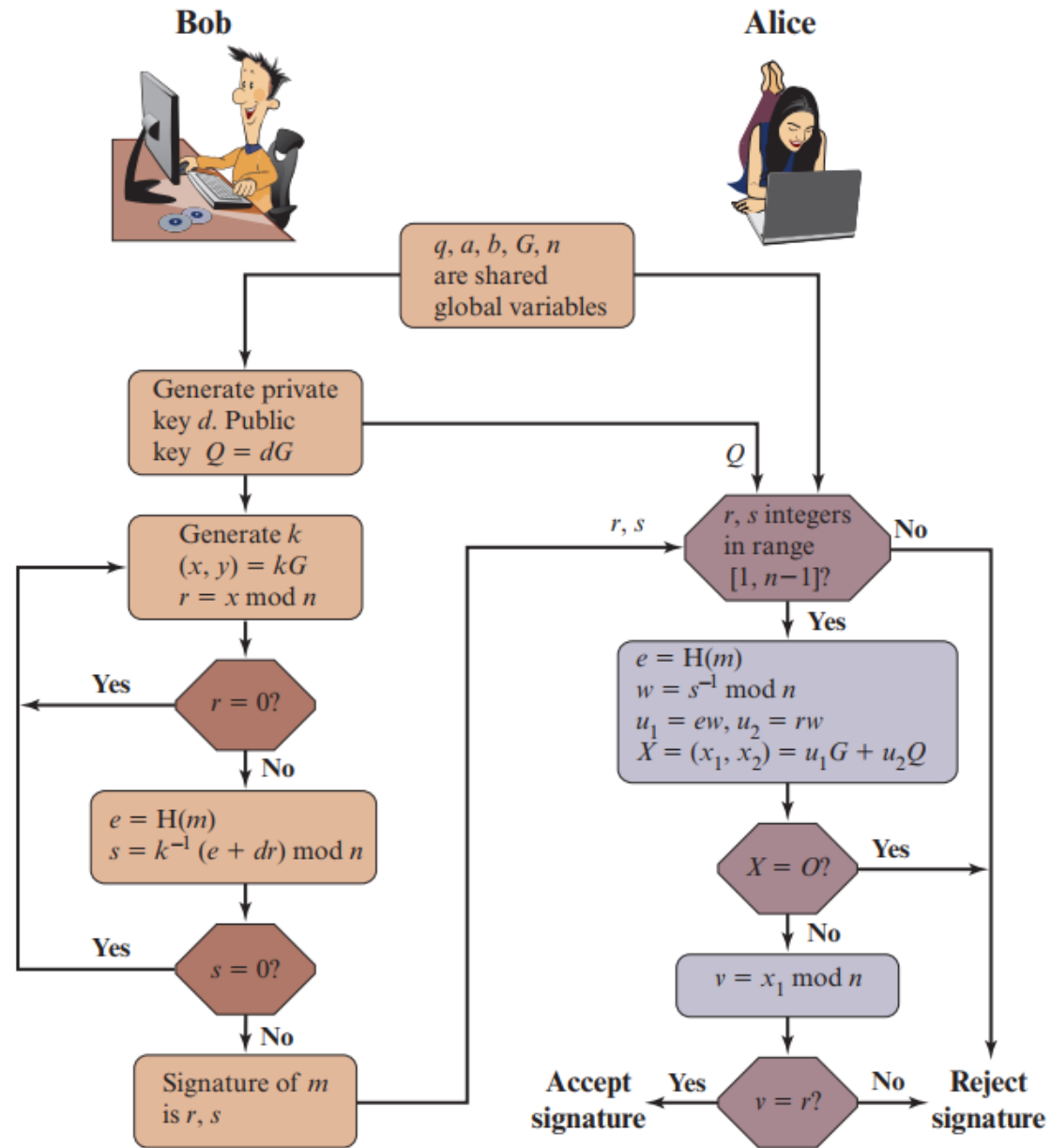
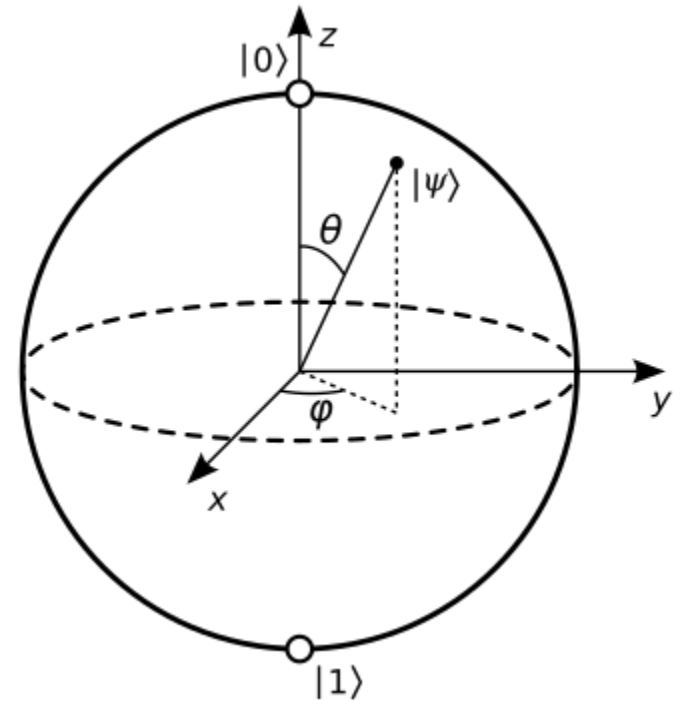


Figure 13.5 ECDSA Signing and Verifying

Post-quantum Cryptography

Quantum computing

- A quantum computer is a computer that exploits quantum mechanical phenomena.



Post-quantum Cryptography

- Post-quantum cryptography is concerned with the development of cryptographic algorithms that are secure against the potential development of quantum computers.
- Quantum computing is based on the representation of information in a form analogous to the behavior of elementary particles in quantum physics.

Post-quantum

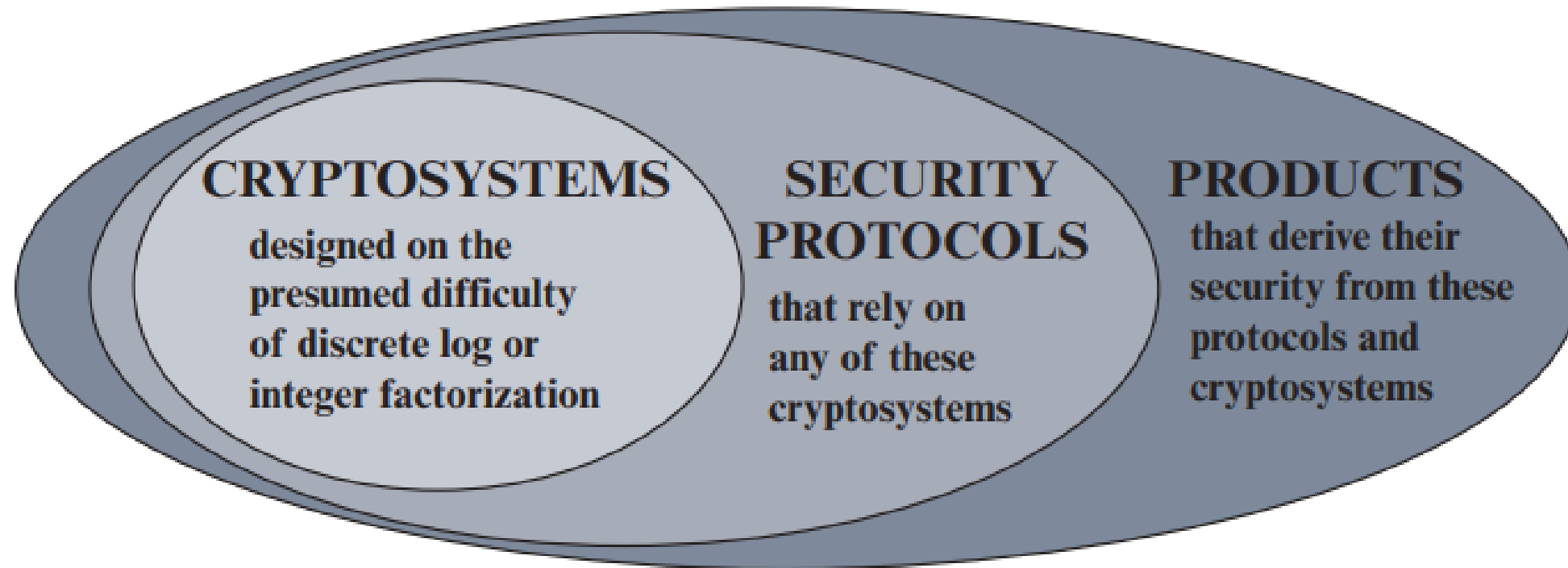


Figure 14.11 Entities Vulnerable to Quantum Computing

Post-quantum

Table 14.6 Impact of Quantum Computing on Common Cryptographic Algorithms

Cryptographic Algorithm	Type	Purpose	Impact from Large-Scale Quantum Computer
AES	Symmetric key	Encryption	Larger key sizes needed
SHA-2, SHA-3	Cryptographic hash	Hash function	Larger output needed
RSA	Asymmetric key	Signature, key establishment	No longer secure
ECDSA, ECDH (elliptic curve cryptography)	Asymmetric key	Signature, key exchange	No longer secure
DSA (finite field cryptography)	Asymmetric key	Signature, key exchange	No longer secure

Post-quantum

Table 14.7 Submissions to NIST Post-Quantum Cryptography Competition

	Signatures	KEM/Encryption	Total
Lattice-based	4	24	28
Code-based	5	19	24
Multivariate	7	6	13
Hash-based	4	—	4
Other	3	10	13
Total	23	59	82