isec

INFORMATION SECURITY

lectures by Ferucio Laurentiu Tiplea

Cryptography Basics

Public Key Cryptography

Prof.dr. Ferucio Laurențiu Țiplea

Fall 2021

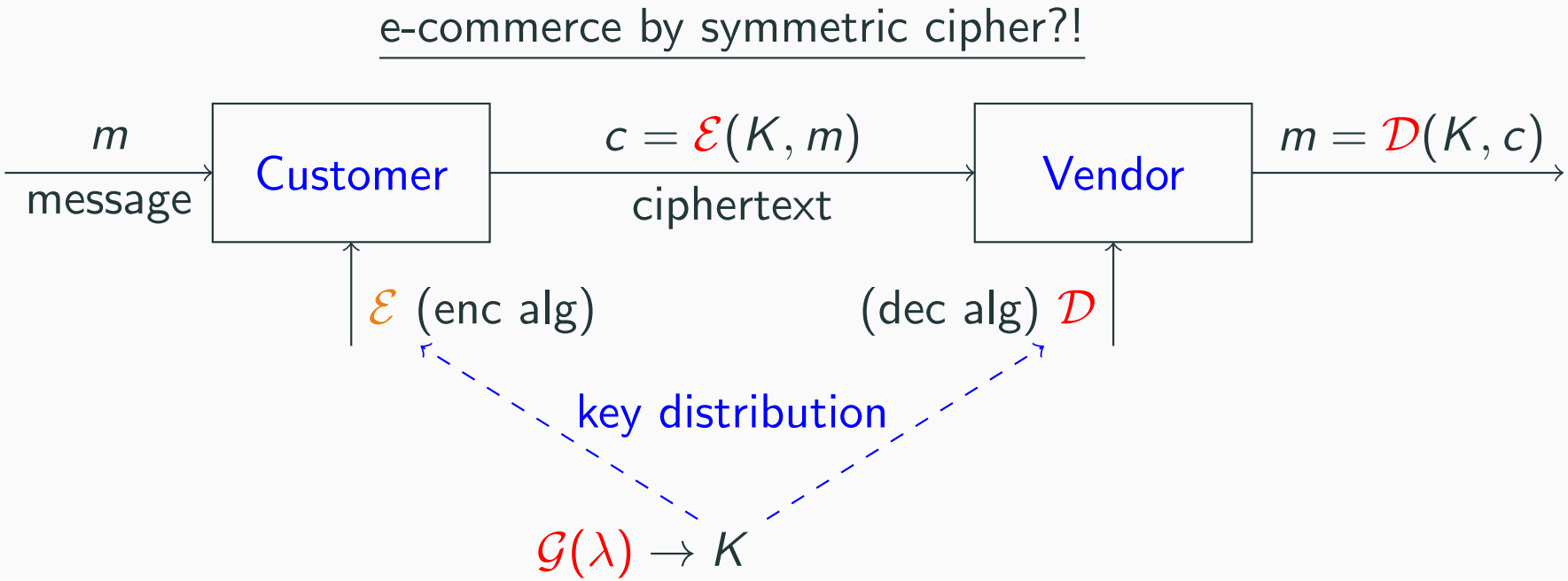
Department of Computer Science
"Alexandru Ioan Cuza" University of Iași
Iași 700506, Romania

e-mail: ferucio.tiplea@uaic.ro

Outline

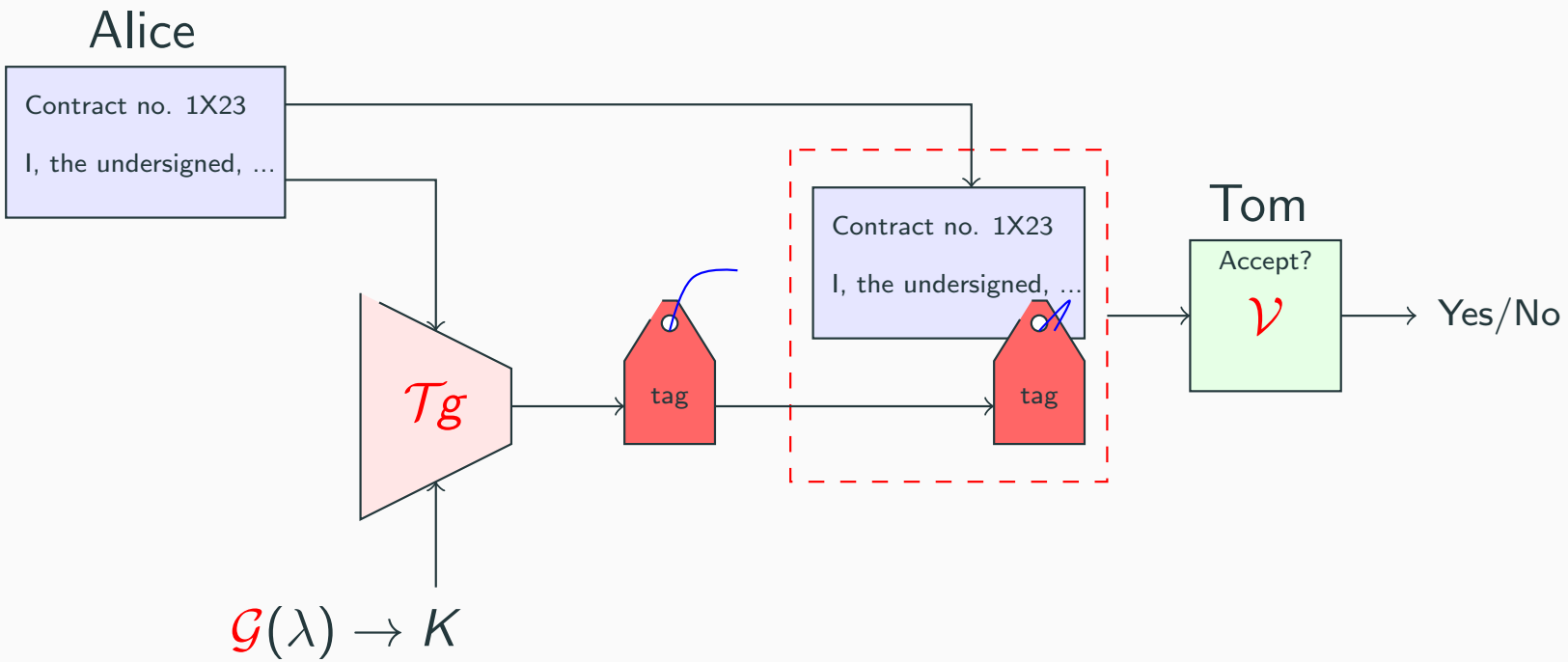
Introduction

Need for secure key distribution



Key distribution: In e-commerce, it is unrealistic to expect initial contact between two parties with no prior acquaintance or postpone their communication long enough for keys to be transmitted by some physical means.

Need for authentication



Authentication: Can Alice use MACs to authenticate (sign!) documents, so that Tom checks them without Alice's authentication key? If Tom knew Alice's key, then Tom would be able to authenticate documents in place of Alice.

Why public key cryptography?

Two main objectives that cannot be achieved by symmetric-key cryptography:

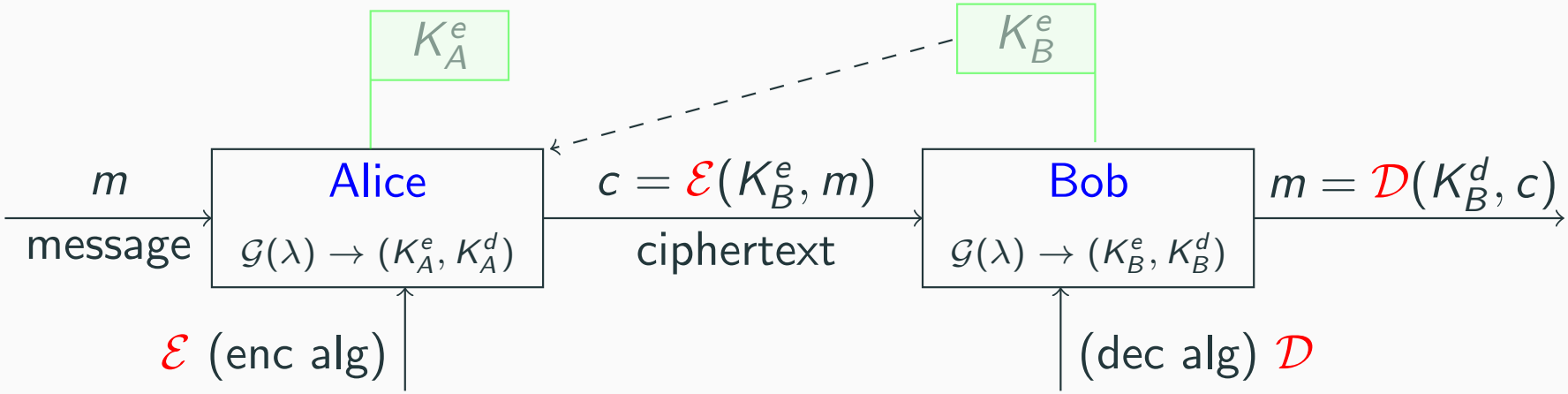
- Need for secure key distribution – A private conversation between two people with no prior acquaintance is a common occurrence in business, and it is unrealistic to expect initial business contacts to be postponed long enough for keys to be transmitted by some physical means;
- Need for authentication – In current business, the validity of contracts is guaranteed by signatures. Is there any purely digital replacement of the handwritten signature?

A bit of history

- 1976: Whitfield Diffie and Martin Hellman, and independently Ralph Merkle, invented public key cryptography to address the two aforementioned deficiencies;
- The first concrete realization of a public key cryptosystem is due to R.C. Merkle and M.E. Hellman in 1978. Unfortunately, this cryptosystem, as well as many other variations of it, have been proved to be insecure;
- Soon after the Merkle-Hellman cryptosystem came the first full-fledged public key cryptosystem, RSA (named after its inventors, R. Rivest, A. Shamir, and L. Adleman). RSA is by far the easiest to understand and implement public key cryptosystem; it gets its security from the difficulty of factorization of very large numbers.

Public key encryption

Public key encryption



$\mathcal{S} = (\mathcal{G}, \mathcal{E}, \mathcal{D})$ – public key cipher / public key encryption (PKE) scheme

Requirements:

1. The public key (K_*^e) is assumed to be openly and widely distributed so that anyone can encrypt messages
2. The secret-key (K_*^d) must be kept private

Security models for PKE

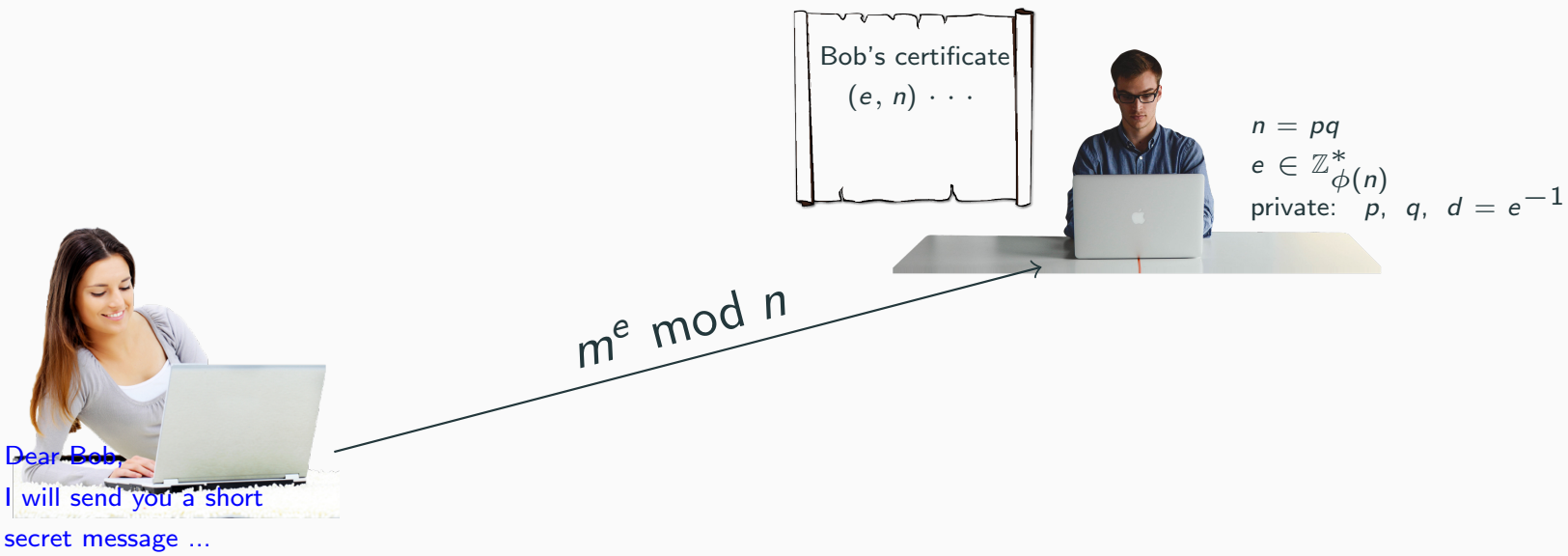
We define and use the same security models based on indistinguishability and non-malleability as in the case of symmetric encryption. However, we draw attention to:

1. No deterministic PKE scheme can be IND-CPA secure – the adversary can consult the encryption oracle any time and,therefore, it can decide what message was encrypted

2. No PKE scheme can achieve perfect security – given a message m and a ciphertext c, and assuming $0 < P(m) < 1$, and unbounded adversary can decide whether c comes from m or not. That is, $P(m|c)$ is 0 or 1. Then, $P(m|c) \neq P(m)$

The RSA PKE scheme

Rivest et al. (1978) proposed the first public key cryptosystem which is still secure and used (it is known as RSA)



RSA encryption and decryption can be done in $\mathcal{O}((\log n)^3)$

Some security issues

- p and q are large numbers randomly generated (512-bit or even larger)
- e is small (fast encryption) but chosen such that $d > \sqrt[4]{n}$ (otherwise, an efficient attack can be mounted – details will be provided later)
- The RSA PKE scheme is not IND-CPA (encrytion is deterministic)
- If one of p, q, $\phi(n)$, or d can be computed in a reasonable time, then the system is completely broken
- If the same n is used by two different users, any one of them can break the other's encryption

Padded RSA

Public key Cryptography Standard #1 (PKCS #1) provides recommendations for implementing the RSA algorithms. The current version is v.2.2 (Oct 2012):

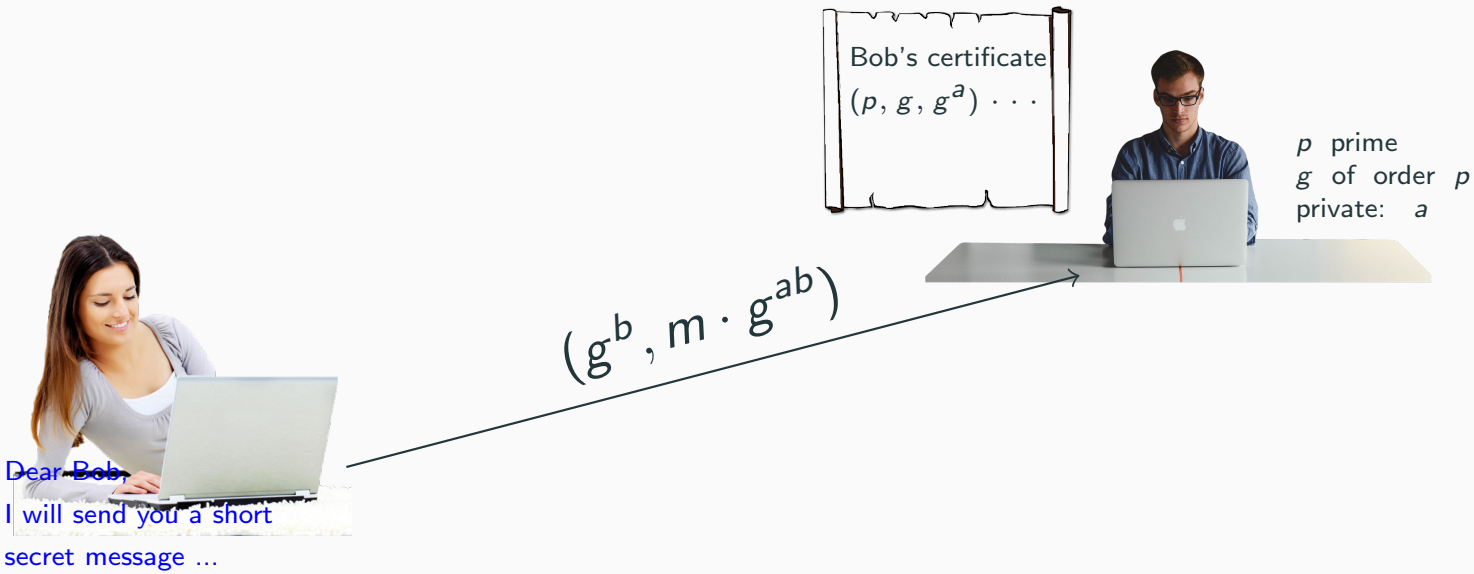
$$0x00 \parallel 0x02 \parallel r \parallel 0x00 \parallel m$$

where m is the message and r is a random string (the length of the above message must be exactly the length of the modulus n, in bytes). Moreover, no byte of r is $0x00$. The second byte ($0x02$) denotes "encryption".

Bleichenbacher (1998) described an attack on PKCS #1, v.1.5, known today as Bleichenbacher's attack. Subsequently, new attacks were found Coron et al. (2000); Izu et al. (2007); Bardou et al. (2012); Böck et al. (2018).

ElGamal PKE scheme

ElGamal (1985) adapted the Diffie-Hellman key-exchange protocol to get a PKE scheme and a digital signature



ElGamal encryption and decryption can be done in $\mathcal{O}((\log n)^3)$

The ElGamal encryption is randomized and IND-CPA, provided that the decisional Diffie-Hellman problem is hard!

Facts

- SKE is significantly faster than PKE
- SKE schemes have lower ciphertext expansion
- Unfortunately, SKC does not provide an adequate key exchange mechanism, but PKC provides!

Use SKE to encrypt messages and PKE to encrypt the secret key for SKE!

Hybrid encryption

1. $\mathcal{S} = (\mathcal{G}, \mathcal{E}, \mathcal{D})$ PKE scheme
2. $\mathcal{S}' = (\mathcal{G}', \mathcal{E}', \mathcal{D}')$ SKE scheme
3. Hybrid encryption with $(\mathcal{S}, \mathcal{S}')$
   - 3.1 $(pk, sk) \leftarrow \mathcal{G}(\lambda)$
   - 3.2 To encrypt m do:
     - $K \leftarrow \mathcal{G}'(\lambda)$
     - $c_1 \leftarrow \mathcal{E}(pk, K)$
     - $c_2 \leftarrow \mathcal{E}'(K, m)$
     - $c = (c_1, c_2)$
   - 3.3 To decrypt c do:
     - $K \leftarrow \mathcal{D}'(sk, c_1)$
     - $m \leftarrow \mathcal{D}(K, c_2)$

Theorem 1

If $\mathcal{S}$ is an IND-CPA secure PKE scheme and $\mathcal{S}'$ is an IND-COA secure SKE scheme, then the HE scheme $(\mathcal{S}, \mathcal{S}')$ is IND-CPA secure.

IND-COA security of the scheme $\mathcal{S}'$ is sufficient because a fresh key is chosen each time a new message is encrypted!

Hybrid encryption is a PKE scheme because the sender and receiver do not share any secret key in advance!

Digital signatures

Digital Signatures

A digital signature scheme is a method of signing a digital message

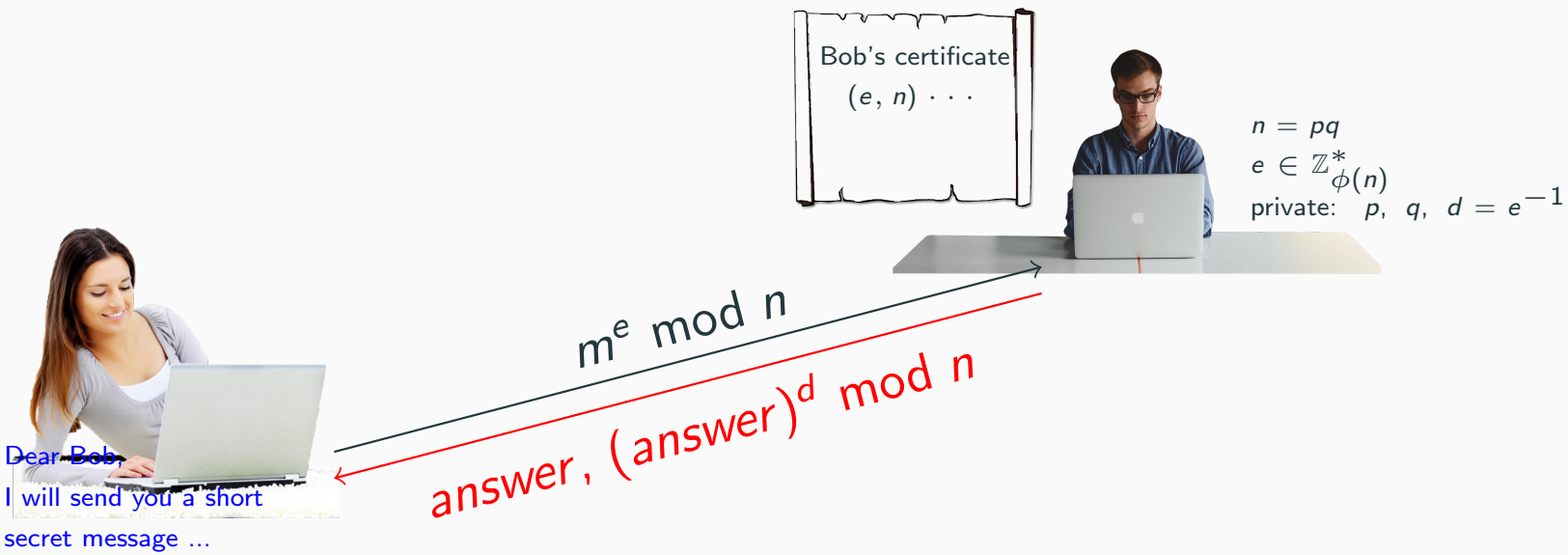
There are several differences between digital signatures and handwritten signatures:

- a digital signature is not attached physically to the message it signs;
- a digital signature depends on the message;
- a digital signature is verified by taking into consideration the message it signs;
- a digital signature scheme should be private, but publicly verifiable.

A digital signature should be "small" so that it can be used on smart cards. Therefore, for long messages, a hash function should be applied first

The RSA signature scheme

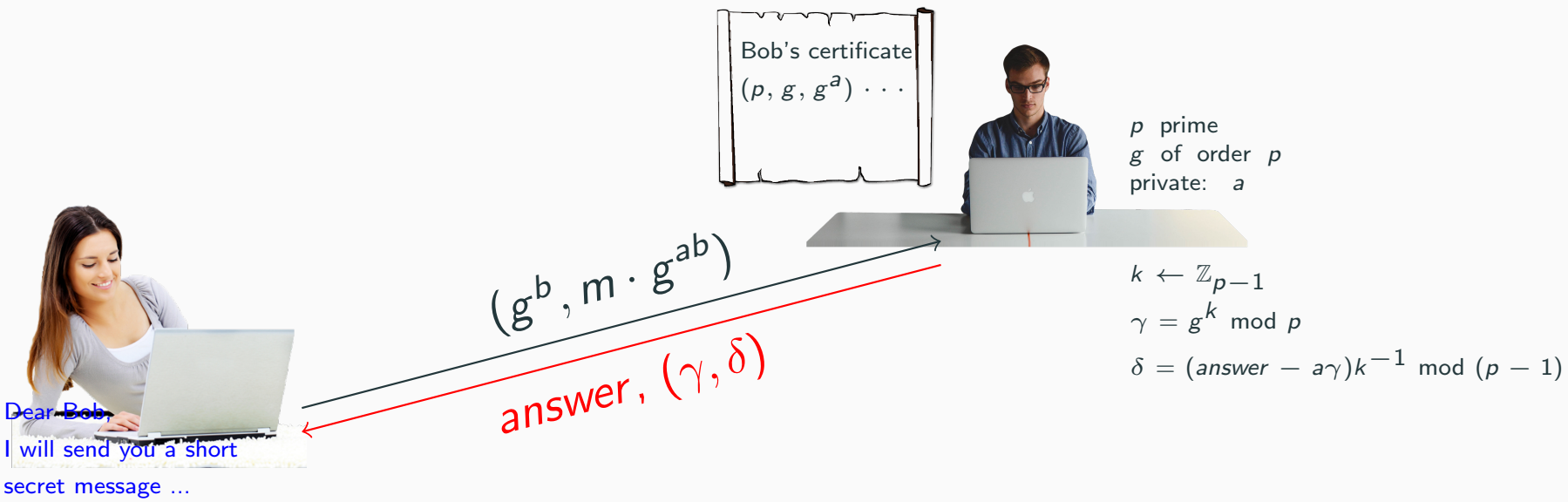
Derived from the RSA encryption scheme: sign with the private key and verify with the public key



Verification: $answer \stackrel{?}{=} ((answer)^d)^e \bmod n$

ElGamal digital signature

In 1985, Taher El Gamal (ElGamal, Elgamal) adapted the Diffie-Hellman key-exchange protocol to get a PKE scheme and a digital signature



Verification: $g^{answer} \stackrel{?}{\equiv} (g^a)^\gamma \gamma^\delta \bmod p$

Digital Signature Standard

- Digital Signature Standard (DSS) is the American standard for digital signatures;
- DSS was proposed by NIST in 1991, and adopted in 1994;
- DSS is a variation of the ElGamal digital signature. This variation is based on the following remark: the prime p in the ElGamal digital signature should be a 512-bit or 1024-bit number in order to ensure security. This fact leads to signatures that are too large to be used on smart cards;
- DSS modifies ElGamal digital signature so that the computations are done in a subgroup $\mathbb{Z}_q$ of $\mathbb{Z}_p^*$ by using an element $\alpha \in \mathbb{Z}_p^*$ of order q.

Elliptic curve cryptography

Elliptic curve cryptography (ECC)

Born Lenstra (1987); Koblitz (1987); Miller (1986)

Goal reduce the size of encryption keys while maintaining the level of security. For instance, a 256-bit key in ECC offers about the same security as a 3072-bit RSA public key

Why Recent research in computational number theory lead to increased security requirements which, in turn, require larger public keys

How Replace the standard group $\mathbb{Z}_n^*$ from number theory with groups of points with integer coordinates on an elliptic curve

Elliptic curves

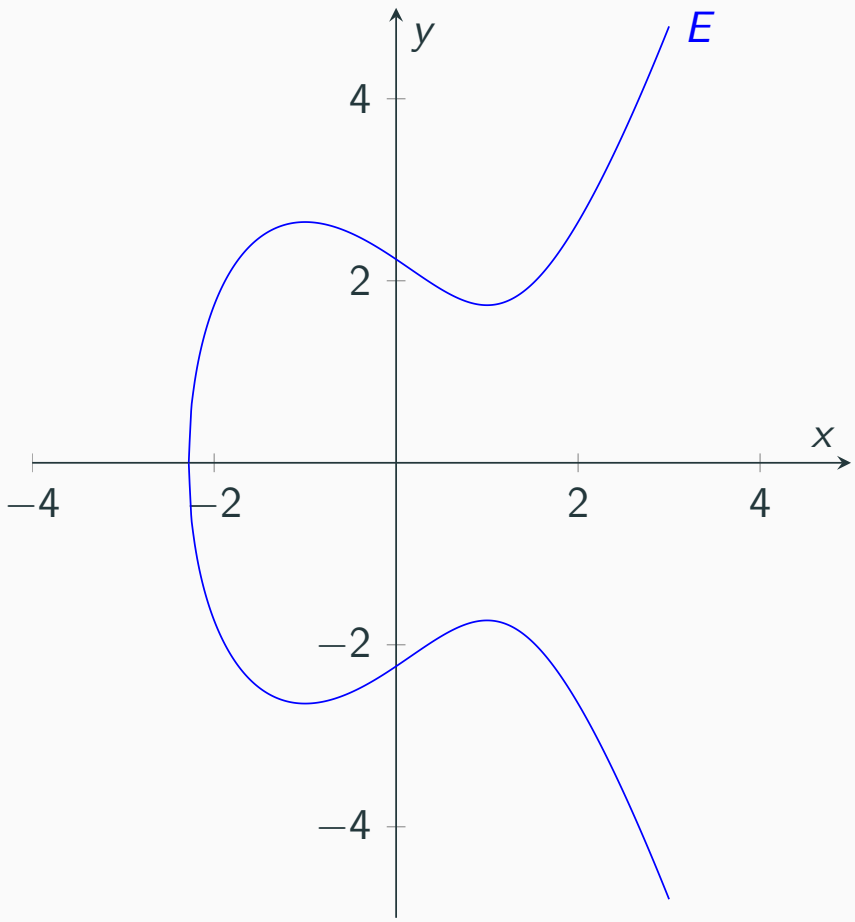
We use elliptic curves (EC) defined by a short Weierstrass equation over finite fields:

$$y^2 = x^3 + Ax + B$$

Example 2

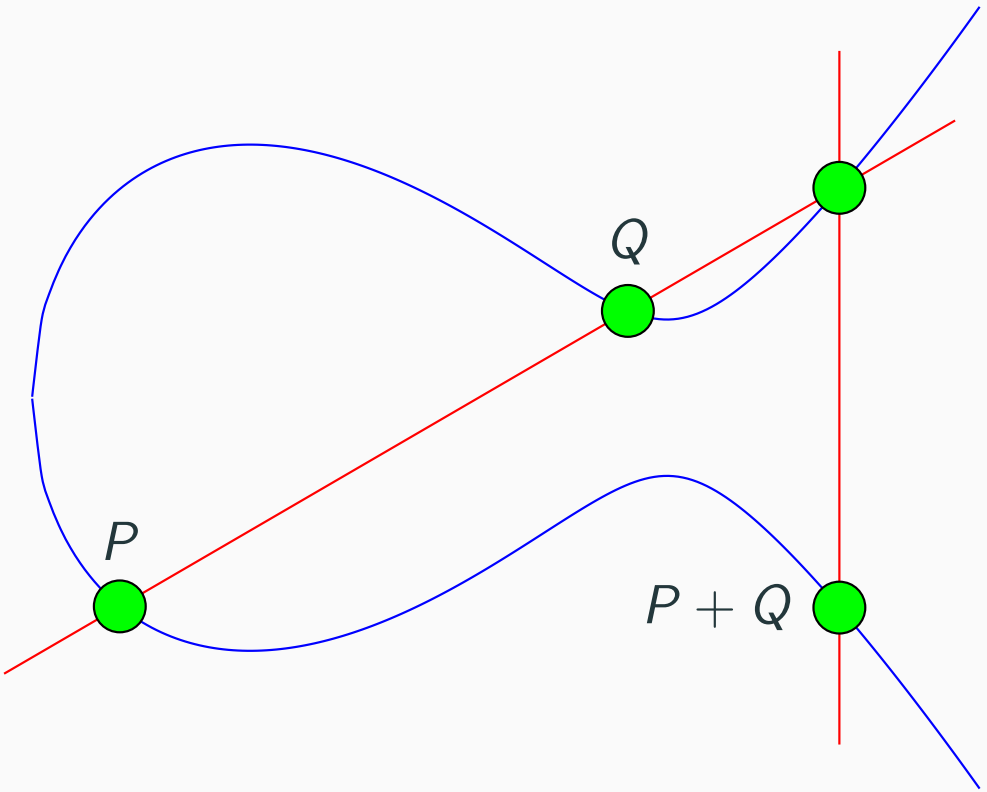
Let $y^2 = x^3 + x + 1$ over $\mathbb{Z}_5$. It gives rise to the following points:

x	y	points
0	± 1	$(0,1)$, $(0,4)$
1	–	–
2	± 1	$(2,1)$, $(2,4)$
3	± 1	$(3,1)$, $(3,4)$
4	± 2	$(4,2)$, $(4,3)$
∞	∞	$\mathcal{O}$



$y^2 = x^3 - 3x + 5$ over $\mathbb{R}$

Addition of points on an EC



Addition of points defines a group algebraic structure

$$mP = \underbrace{P + P + \cdots + P}_{m \text{ times}}$$

Discrete logarithm problem on EC

Elliptic Curve Discrete Logarithm Problem (ECDLP)

Instance: Elliptic curve E and two points P and mP on E

Question: Compute m

Facts:

1. No sub-exponential algorithms are known to solve general-purpose ECDLP (but they are known for DLP over finite fields!)
2. All ECC schemes are public key and are based on ECDLP. Most of them are ECC versions of classical schemes
3. The absence of sub-exponential algorithms to solve the general-purpose ECDLP implies that the ECDLP-based cryptographic schemes can use smaller fields than those based on DLP on finite fields. As a result, the cryptographic keys have smaller sizes

Key length comparison

NIST recommendation for symmetric and public keys (Barker (2020)):

Symmetric	Factoring-based (RSA)	Finite field based (DSA, DH, MQV)	EC
80	1024	1024	160
112	2048	2048	224
128	3072	3072	258
192	7680	7680	384
256	15360	15360	512

All key sizes are in bits

Public key certificates

X.509 certification scheme

How does Alice obtain Bob's public key?

In general, how can public keys for encryption or verification of digital signatures be distributed in an authenticated way?

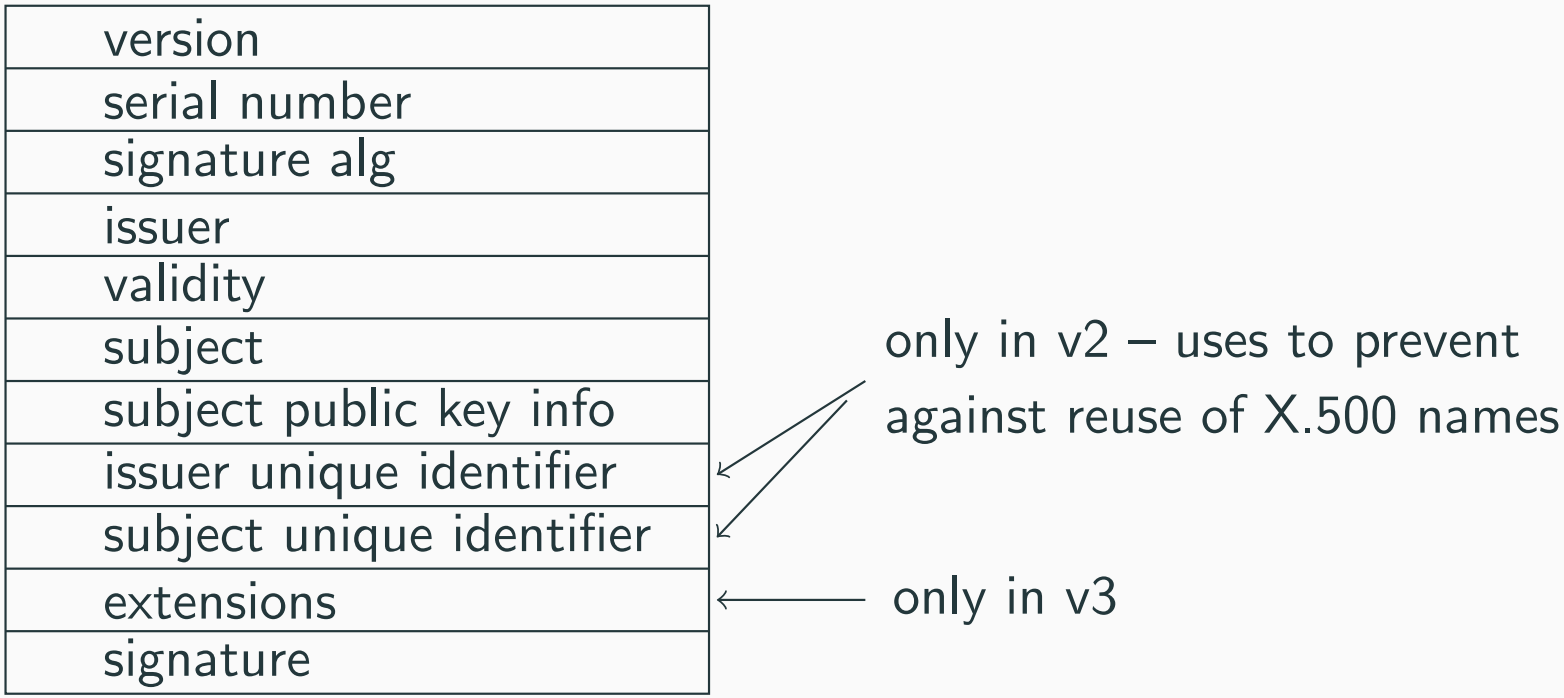
- Kohnfelder (1978) introduced in his B.Sc. thesis the concepts of certificate and certificate revocation list
- A certificate binds an identity to a public key
- X.509 is a standard for defining public key certificates
- X.509 certificates are used in many Internet protocols

A public key infrastructure (PKI) is a collection of tools (roles, policies, hardware and/or software tools) needed to create, manage, distribute, use, store and revoke digital certificates and manage public key encryption.

X.509 certificate

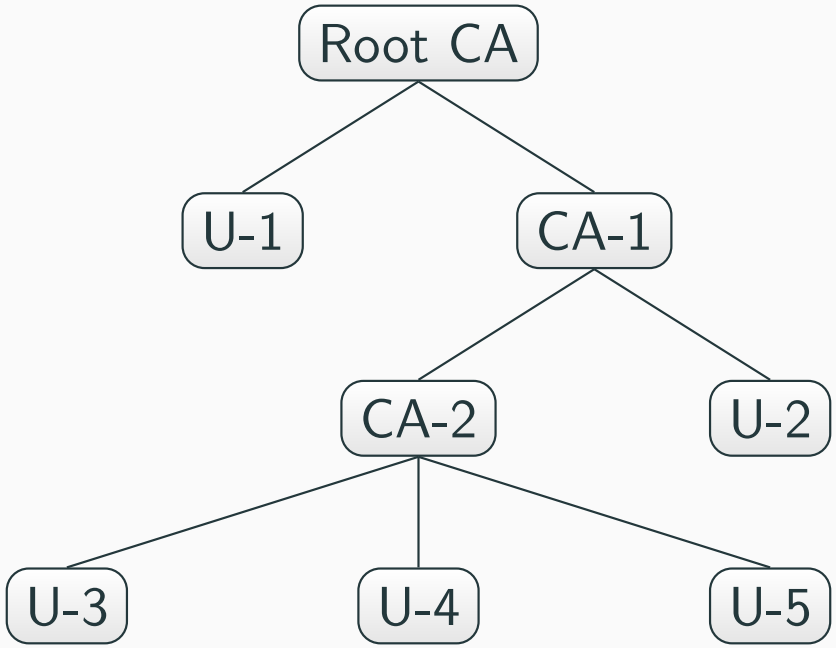
Because a certificate binds an identity to a public key, it must be authenticated. This is done through digital signatures of trusted authorities, called certificate/certification authorities (CAs).

X.509 certificate v1 / v2 / v3



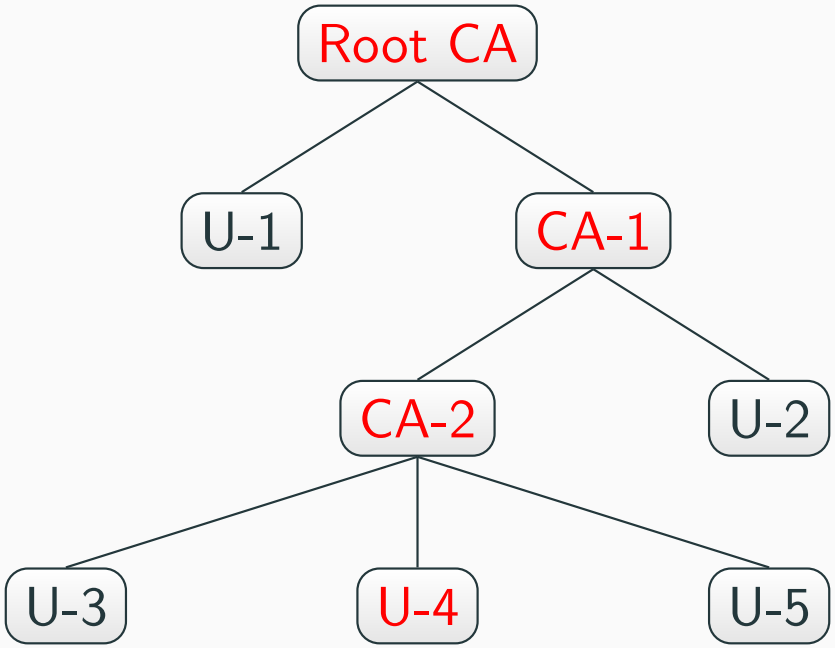
CA hierarchy

Many practical reasons lead to the need for more CAs, interconnected hierarchically as in the diagram below.



Certificate chains

A certificate chain is a sequence of certificates that corresponds to a path in the CAs hierarchy, starting with a leaf but not necessarily going to the root. The last certificate in a certificate chain is called a trust anchor for the first certificate.

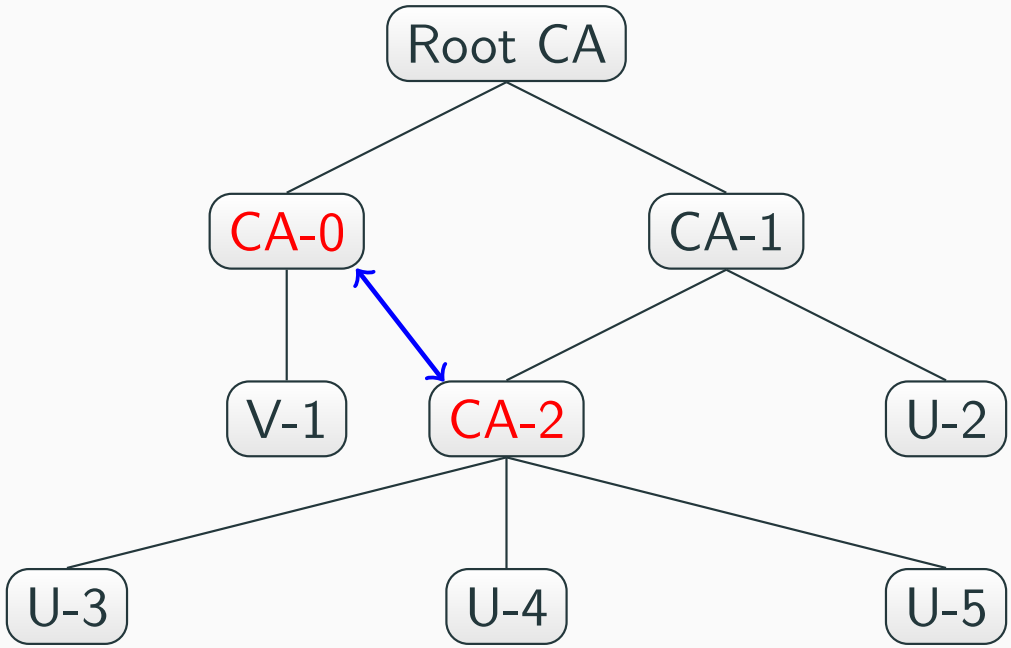


Root CA signs CA-1's certificate,
CA-1 signs CA-2's certificate,
CA-2 signs U-4's certificate.

The authenticity verification is in reverse order.

Cross certification

Cross certification enables entities in one public key infrastructure (PKI) to trust entities in another PKI. Cross certification requires that each CA issue a certificate to the other to establish the relationship in both directions. The trust path is not hierarchal (neither of the governing CAs is subordinate to the other)



PKI issues

1. Difficulty in retrieving keys and certificates
2. Questionable value of certified key representations
3. Certificate processing complexity
4. Costly certificates
5. Problematic cross-domain trust management
6. Naming semantics
7. Use with insecure clients
8. Privacy compromises

PKI: usability studies

1. Why Johny Can't Encrypt: A Usability Case Study of PGP 5.0, by A. Whitten and J.D. Tyger, 1999
2. Johny 2: A User Test of Key Continuity Management with S/MIME and Outlook Express, by S.I. Garfinkel and R.C. Miller, 2005
3. Why Johny Still Can't Encrypt: Evaluating the Usability of Email Encryption Software, by S. Sheng, L. Broderick, C.A. Koranda, and J.L. Hyland, 2006
4. Why (Special Agent) Johnny (Still) Can't Encrypt: A Security Analysis of the APCO Project 25 Two-Way Radio System, by S. Clark, T. Goodspeed, P. Metzger, Z. Wasserman, K. Xu, and M. Blaze, 2011

References

Bardou, R., Focardi, R., Kawamoto, Y., Simionato, L., Steel, G., and Tsay, J.-K. (2012). Efficient padding oracle attacks on cryptographic hardware. In Safavi-Naini, R. and Canetti, R., editors, *Advances in Cryptology – CRYPTO 2012*, pages 608–625, Berlin, Heidelberg. Springer Berlin Heidelberg.

Barker, E. (2020). Recommendation for key management: Part 1 – general. NIST Pubs 800-57 Rev. 5, NIST.

Bleichenbacher, D. (1998). Chosen ciphertext attacks against protocols based on the rsa encryption standard pkcs #1. In Krawczyk, H., editor, *Advances in Cryptology — CRYPTO '98*, pages 1–12, Berlin, Heidelberg. Springer Berlin Heidelberg.

Böck, H., Somorovsky, J., and Young, C. (2018). Return of bleichenbacher's oracle threat (robot). In *Proceedings of the 27th USENIX Conference on Security Symposium*, SEC'18, page 817–832, USA. USENIX Association.

Coron, J.-S., Joye, M., Naccache, D., and Paillier, P. (2000). New attacks on pkcs#1 v1.5 encryption. In Preneel, B., editor, *Advances in Cryptology — EUROCRYPT 2000*, pages 369–381, Berlin, Heidelberg. Springer Berlin Heidelberg.

References (cont.)

ElGamal, T. (1985). A public key cryptosystem and a signature scheme based on discrete logarithms. In Blakley, G. R. and Chaum, D., editors, *Advances in Cryptology*, pages 10–18, Berlin, Heidelberg. Springer Berlin Heidelberg.

Izu, T., Takenaka, M., and Shimoyama, T. (2007). Analysis on bleichenbacher's forgery attack. In *The Second International Conference on Availability, Reliability and Security (ARES'07)*, pages 1167–1174.

Koblitz, N. (1987). Elliptic curve cryptosystems. *Mathematics of Computation*, 48(177):203–209.

Kohnfelder, L. (1978). Towards a practical public-key cryptosystem.

Lenstra, H. J. (1987). Fcatoring integers with elliptic curves. *The Annals of Mathematics*, 126(3):649–673.

Miller, V. S. (1986). Use of elliptic curves in cryptography. In Williams, H. C., editor, *Advances in Cryptology — CRYPTO '85 Proceedings*, pages 417–426, Berlin, Heidelberg. Springer Berlin Heidelberg.

Rivest, R. L., Shamir, A., and Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. *Commun. of the ACM*, 21(2):120–126.