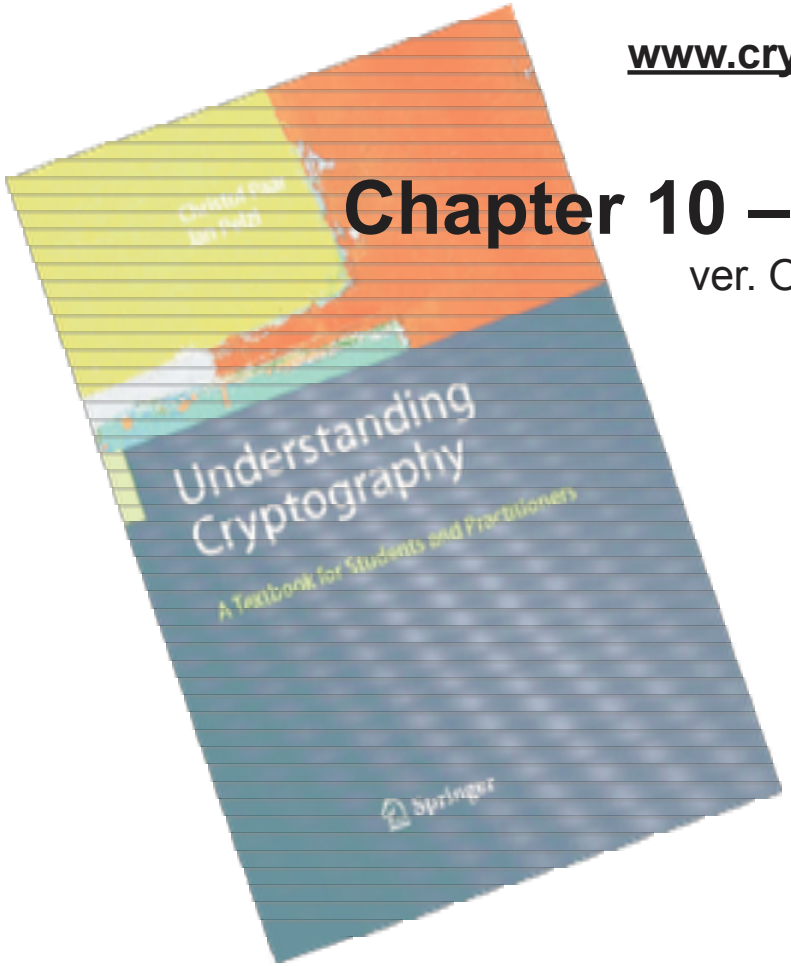
Understanding Cryptography – A Textbook for Students and Practitioners

by Christof Paar and Jan Pelzl

www.crypto-textbook.com



Chapter 10 – Digital Signatures

ver. October 29, 2009

These slides were prepared by Georg Becker, Christof Paar and Jan Pelzl
and modified by Sam Bowne

Some legal stuff (sorry): Terms of Use

- The slides can used free of charge. All copyrights for the slides remain with Christof Paar and Jan Pelzl.
- The title of the accompanying book “Understanding Cryptography” by Springer and the author’s names must remain on each slide.
- If the slides are modified, appropriate credits to the book authors and the book title must remain within the slides.
- It is not permitted to reproduce parts or all of the slides in printed form whatsoever without written consent by the authors.

Contents of this Chapter

10.1.2 Principles of Digital Signatures

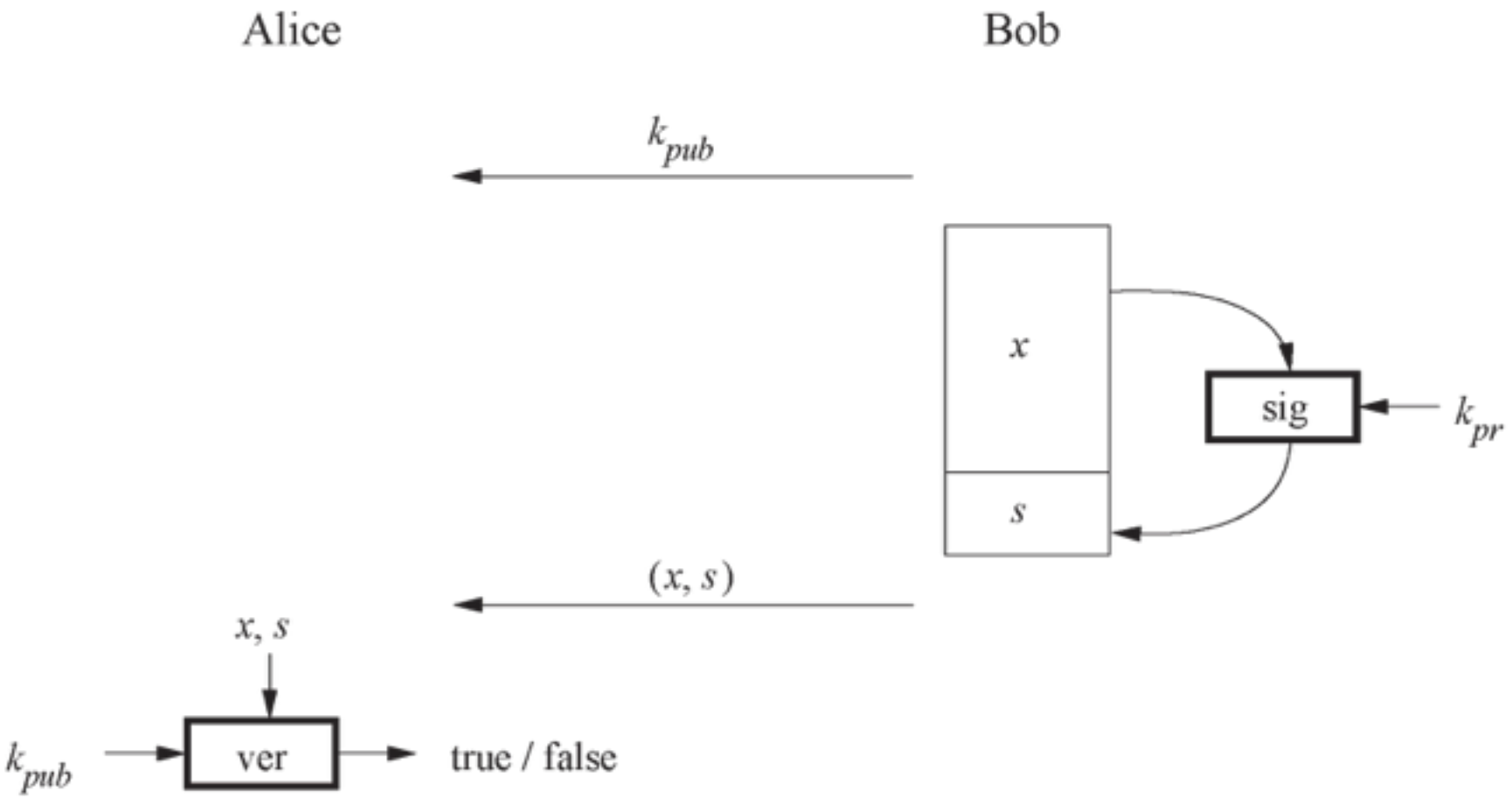
Motivation

- Alice orders a pink car from the car salesman Bob
- After seeing the pink car, Alice states that she never ordered it:
- How can Bob prove to a judge that Alice ordered a pink car? (And that he did not fabricate the order himself)

$\Rightarrow$ Symmetric cryptography fails because both Alice and Bob can be malicious

$\Rightarrow$ Can be achieved with public-key cryptography

Basic Principle of Digital Signatures



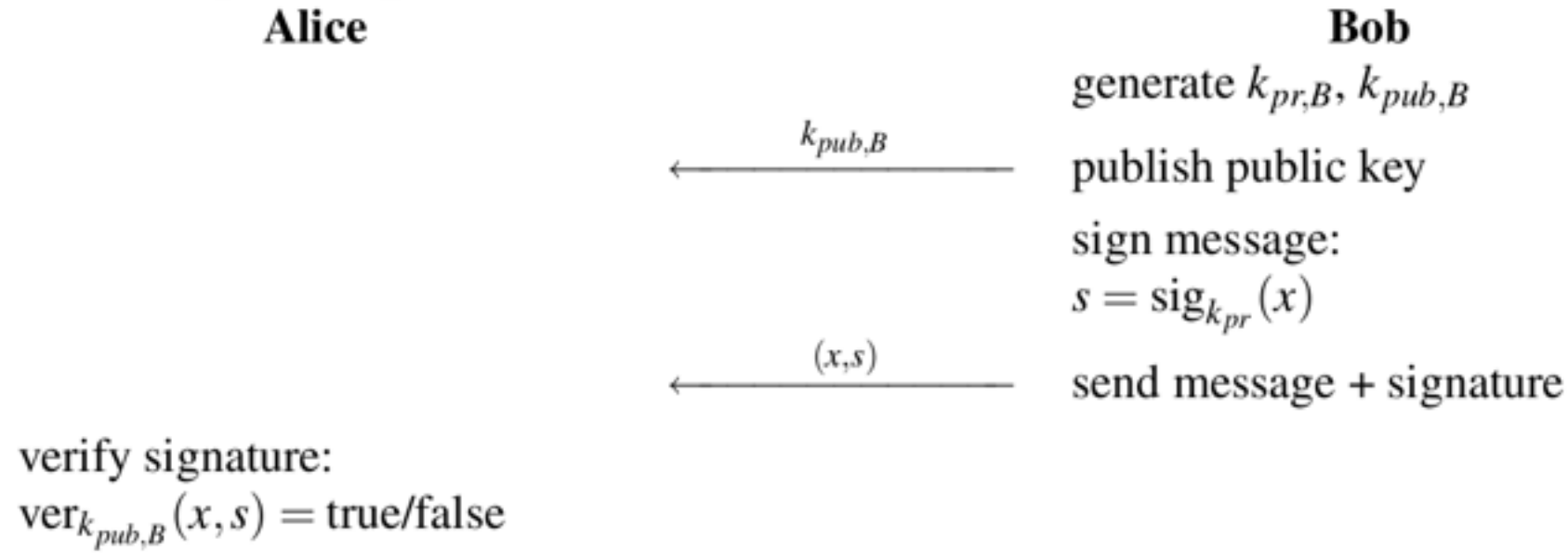
Main idea

- For a given message *x*, a digital signature is appended to the message (just like a conventional signature)
- Only the person with the private key should be able to generate the signature
- The signature must change for every document

$\Rightarrow$ The signature is made from the message *x* and the private key

$\Rightarrow$ The public key and the message *x* are the inputs to the verification function

Basic Digital Signature Protocol



10.1.3 Security Objectives

Core Security Objectives

1. **Confidentiality***:* Information is kept secret from all but authorized parties.
2. **Integrity:** Ensures that a message has not been modified in transit.
3. **Message Authentication:** Ensures that the sender of a message is authentic. An alternative term is data origin authentication.
4. **Non-repudiation:** Ensures that the sender of a message can not deny the creation of the message. (c.f. order of a pink car)

Additional Security Objectives

5. **Identification/entity authentication:** Establishing and verification of the identity of an entity, e.g. a person, a computer, or a credit card.
6. **Access control:** Restricting access to the resources to privileged entities.
7. **Availability:** The electronic system is reliably available.
8. **Auditing:** Provides evidences about security relevant activities, e.g., by keeping logs about certain events.
9. **Physical security:** Providing protection against physical tampering and/or responses to physical tampering attempts
10. **Anonymity:** Providing protection against discovery and misuse of identity.

1

10.2 The RSA Digital Signature scheme

Schoolbook RSA Digital Signature

RSA Keys

- Bob's private key: $k_{pr} = (d)$
- Bob's public key: $k_{pub} = (n, e)$

Basic RSA Digital Signature Protocol

Alice		Bob
		$k_{pr} = d$, $k_{pub} = (n, e)$
	$\xleftarrow{(n,e)}$	
		compute signature: $s = \text{sig}_{k_{pr}}(x) \equiv x^d \bmod n$
	$\xleftarrow{(x,s)}$	

verify: $\text{ver}_{k_{pub}}(x, s)$

$x' \equiv s^e \bmod n$

$$x' \begin{cases} \equiv x \bmod n & \Longrightarrow \text{valid signature} \\ \not\equiv x \bmod n & \Longrightarrow \text{invalid signature} \end{cases}$$

The RSA Signature Protocol

Basic RSA Digital Signature Protocol

Alice | **Bob**

$k_{pr} = d, k_{pub} = (n, e)$

(n,e) ⟵

compute signature:

$s = \text{sig}_{k_{pr}}(x) \equiv x^d \bmod n$

(x,s) ⟵

verify: $\text{ver}_{k_{pub}}(x, s)$

$x' \equiv s^e \bmod n$

$$x' \begin{cases} \equiv x \bmod n & \Longrightarrow \text{valid signature} \\ \not\equiv x \bmod n & \Longrightarrow \text{invalid signature} \end{cases}$$

Security and Performance of the RSA Signature Scheme

Security:

The same constrains as RSA encryption: n needs to be at least 1024 bits to provide a security level of 80 bit.

$\Rightarrow$ The signature, consisting of s, needs to be at least 1024 bits long

Performance:

The signing process is an exponentiation with the private key and the verification process an exponentiation with the public key e.

$\Rightarrow$ Signature verification is very efficient as a small number can be chosen for the public key.

Attacks on the RSA Signature Scheme

Counterfeit public key:

Attacker publishes a public key under someone else's name

Countermeasure: digital certificates and PKI

Factoring n:

Attacker calculates private key from public key \

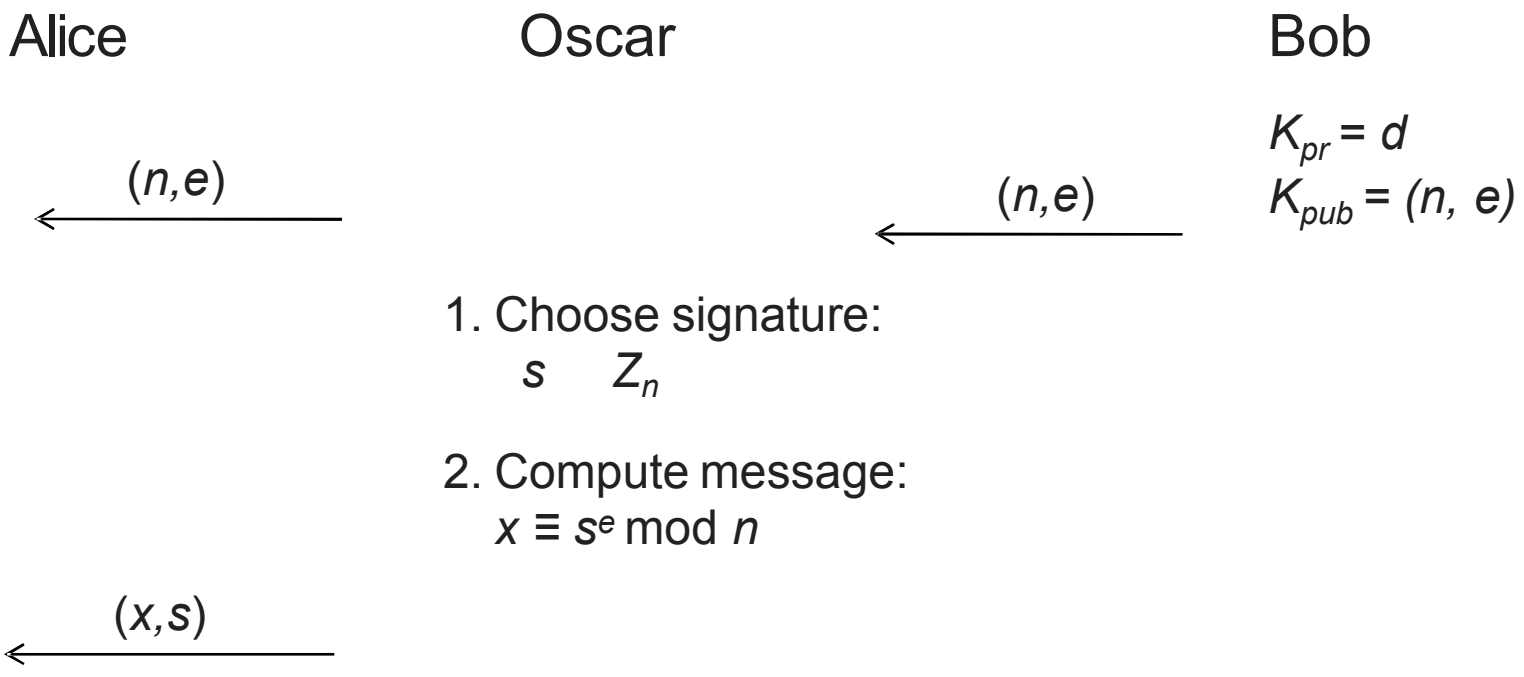
Countermeasure: **n** > 1024 bits

Existential forgery:

Attacker starts from a signature and crafts a message that matches it

Countermeasure: padding

Existential Forgery Attack against RSA Digital Signature



Verification:
$s^e \equiv x' \bmod n$

since $s^e = (x^d)^e \equiv x \bmod n$
→ Signature is valid

Existential Forgery and Padding

- An attacker can generate valid message-signature pairs (x,s)
- But an attack can only choose the signature s and NOT the message x

$\Rightarrow$ Attacker cannot generate messages like „Transfer $1000 into Oscar‘s account“

Formatting the message x according to a *padding scheme* can be used to make sure that an attacker cannot generate valid (x,s) pairs.

Probabilistic Signature Standard (PSS)

- **Formatting rule** distinguishes between valid and invalid messages
- The padding includes a random salt and a hash of the message
    - So signing the same message twice results in different padding
- Existential forgery results in a random message which is very unlikely to contain a valid hash value

Encoding for the EMSA Probabilistic Signature Scheme

Let $|n|$ be the size of the RSA modulus in bits. The encoded message EM has a length $\lceil (|n|-1)/8 \rceil$ bytes such that the bit length of EM is at most $|n|-1$ bit.

1. Generate a random value $salt$.
2. Form a string M' by concatenating a fixed padding $padding_1$, the hash value $mHash = h(M)$ and $salt$.
3. Compute a hash value H of the string M'.
4. Concatenate a fixed padding $padding_2$ and the value $salt$ to form a data block DB.
5. Apply a mask generation function MGF to the string M' to compute the mask value $dbMask$. In practice, a hash function such as SHA-1 is often used as MGF.
6. XOR the mask value $dbMask$ and the data block DB to compute $maskedDB$.
7. The encoded message EM is obtained by concatenating $maskedDB$, the hash value H and the fixed padding bc.

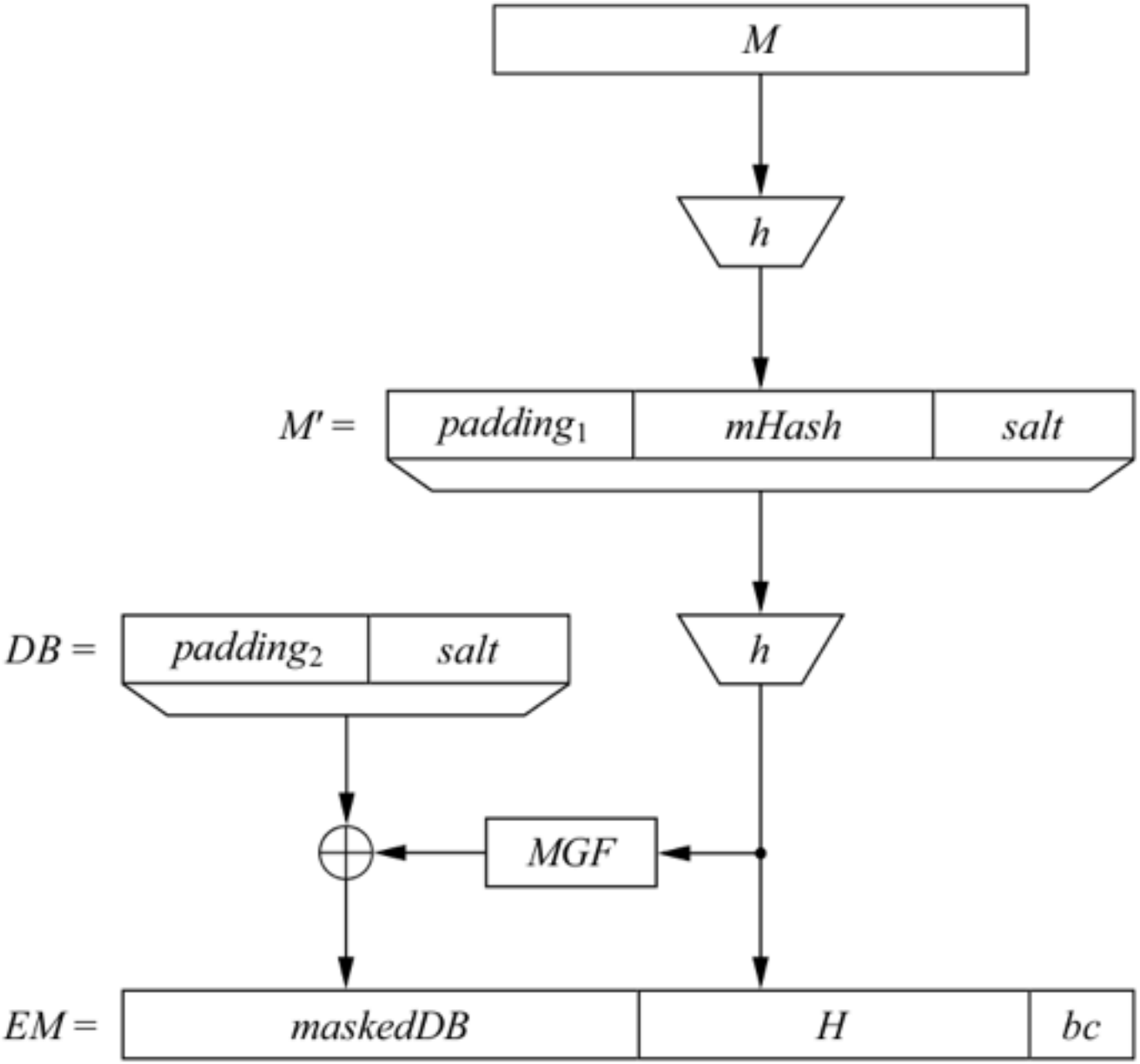


Fig. 10.2 Principle of EMSA-PSS encoding

4

10.3 The Elgamal Digital Signature Scheme

Elgamal Signatures

- Generate a key pair
- Choose a random ephemeral key
- Calculate a signature using the private key, the ephemeral key, and the message
- Receiver can verify the signature from the public key

Schoolbook Elgamal Digital Signature

Alice

Bob

1. choose $p = 29$
2. choose $\alpha = 2$
3. choose $d = 12$
4. $\beta = \alpha^d \equiv 7 \bmod 29$

$\xleftarrow{(p,\alpha,\beta)=(29,2,7)}$

compute signature for message $x = 26$:
choose $k_E = 5$, note that $\gcd(5, 28) = 1$
$r = \alpha^{k_E} \equiv 2^5 \equiv 3 \bmod 29$
$s = (x - d\,r)\,k_E^{-1} \equiv (-10) \cdot 17 \equiv 26 \bmod 28$

$\xleftarrow{(x,(r,s))=(26,(3,26))}$

verify:
$t = \beta^r \cdot r^s \equiv 7^3 \cdot 3^{26} \equiv 22 \bmod 29$
$\alpha^x \equiv 2^{26} \equiv 22 \bmod 29$
$t \equiv \alpha^x \bmod 29 \Longrightarrow$ valid signature

Schoolbook Elgamal Digital Signature

Key Generation for Elgamal Digital Signature

1. Choose a large prime p.
2. Choose a primitive element α of $\mathbb{Z}_p^*$ or a subgroup of $\mathbb{Z}_p^*$.
3. Choose a random integer $d \in \{2, 3, \ldots, p-2\}$.
4. Compute $\beta = \alpha^d \bmod p$.

The public key is now formed by $k_{pub} = (p, \alpha, \beta)$, and the private key by $k_{pr} = d$.

Schoolbook Elgamal Digital Signature

Elgamal Signature Generation

1. Choose a random ephemeral key $k_E \in \{0, 1, 2, \ldots, p-2\}$ such that $\gcd(k_E, p-1) = 1$.
2. Compute the signature parameters:

$$r \equiv \alpha^{k_E} \bmod p,$$
$$s \equiv (x - d \cdot r)\, k_E^{-1} \bmod p - 1.$$

Elgamal Signature Verification

1. Compute the value

$$t \equiv \beta^r \cdot r^s \bmod p$$

2. The verification follows from:

$$t \begin{cases} \equiv \alpha^x \bmod p & \Longrightarrow \text{valid signature} \\ \not\equiv \alpha^x \bmod p & \Longrightarrow \text{invalid signature} \end{cases}$$

Security

- Verifier must have correct public key
- **p** must be 1024 bits or longer to make the DLP (Discrete Logarithm Problem) sufficiently difficult
- Signer must not re-use the ephemeral key
  - Otherwise attacker can easily find the private key
  - Attacker has two equations with two unknowns
    - $\mathbf{k_E}$ and **d**

$$s_1 \equiv (x_1 - d\,r)k_E^{-1} \bmod p - 1$$
$$s_2 \equiv (x_2 - d\,r)k_E^{-1} \bmod p - 1$$

Existential Forgery Attack

- Similar to RSA attack
- Attacker can generate a valid signed message, but the plaintext is scrambled
- Hashing the message before signing it makes this attack impossible

10.4 The Digital Signature Algorithm (DSA)

Digital Signature Algorithm (DSA)

- Federal US Government standard for digital signatures (DSS)
- Proposed by the National Institute of Standards and Technology (NIST)
- DSA is based on the Elgamal signature scheme
- Signature is only 320 bits long
- Signature verification is slower compared to RSA
- Far more popular than Elgamal signature, which are rarely used

Digital Signature Algorithm (DSA)

- Uses two cyclic groups
- In 1024-bit version:
  - Larger group is near 10**1024
  - Smaller group is near 2**160
- Longer keys are possible in the standard

Bit lengths of important parameters of DSA

p	q	Signature
1024	160	320
2048	224	448
3072	256	512

The Digital Signature Algorithm (DSA)

Key Generation for DSA

1. Generate a prime p with $2^{1023} < p < 2^{1024}$.
2. Find a prime divisor q of $p-1$ with $2^{159} < q < 2^{160}$.
3. Find an element α with $\text{ord}(\alpha) = q$, i.e., α generates the subgroup with q elements.
4. Choose a random integer d with $0 < d < q$.
5. Compute $\beta \equiv \alpha^d \bmod p$.

The keys are now:

$k_{pub} = (p, q, \alpha, \beta)$

$k_{pr} = (d)$

The Digital Signature Algorithm (DSA)

DSA Signature Generation

1. Choose an integer as random ephemeral key k_E with $0 < k_E < q$.
2. Compute $r \equiv (\alpha^{k_E} \bmod p) \bmod q$.
3. Compute $s \equiv (SHA(x) + d \cdot r)\, {k_E}^{-1} \bmod q$.

The Digital Signature Algorithm (DSA)

DSA Signature Verification

1. Compute auxiliary value $w \equiv s^{-1} \bmod q$.
2. Compute auxiliary value $u_1 \equiv w \cdot SHA(x) \bmod q$.
3. Compute auxiliary value $u_2 \equiv w \cdot r \bmod q$.
4. Compute $v \equiv (\alpha^{u_1} \cdot \beta^{u_2} \bmod p) \bmod q$.
5. The verification $ver_{k_{pub}}(x, (r, s))$ follows from:

$$v \begin{cases} \equiv r \bmod q \implies \text{valid signature} \\ \not\equiv r \bmod q \implies \text{invalid signature} \end{cases}$$

Security of DSA

To solve the discrete logarithm problem in p the powerful index calculus method can be applied. But this method cannot be applied to the discrete logarithm problem of the subgroup q. Therefore q can be smaller than p.

p	q	hash output (min)	security levels
1024	160	160	80
2048	224	224	112
3072	256	256	128

Standardized parameter bit lengths and security levels for the DSA

Security of DSA

Must not re-use ephemeral key

Just like Elgamal signatures

10.5 The Elliptic Curve Digital Signature Algorithm (ECDSA)

Elliptic Curve Digital Signature Algorithm (ECDSA)

- Based on Elliptic Curve Cryptography (ECC)
- Bit lengths in the range of 160-256 bits provide security equivalent to 1024-3072 bit RSA (80-128 bit symmetric security level)
- One signature consists of two points, hence the signature is twice the used bit length (i.e., 320-512 bits for 80-128 bit security level).
- The shorter bit length of ECDSA often result in shorter processing time

Elliptic Curve Digital Signature Algorithm (ECDSA)

Key Generation for ECDSA

1. Use an elliptic curve E with
   - modulus p
   - coefficients a and b
   - a point A which generates a cyclic group of prime order q
2. Choose a random integer d with $0 < d < q$.
3. Compute $B = dA$.

The keys are now:

$k_{pub} = (p, a, b, q, A, B)$

$k_{pr} = (d)$

Elliptic Curve Digital Signature Algorithm (ECDSA)

ECDSA Signature Generation

1. Choose an integer as random ephemeral key k_E with $0 < k_E < q$.
2. Compute $R = k_E A$.
3. Let $r = x_R$.
4. Compute $s \equiv (h(x) + d \cdot r)\, {k_E}^{-1} \bmod q$.

ECDSA Signature Verification

1. Compute auxiliary value $w \equiv s^{-1} \bmod q$.
2. Compute auxiliary value $u_1 \equiv w \cdot h(x) \bmod q$.
3. Compute auxiliary value $u_2 \equiv w \cdot r \bmod q$.
4. Compute $P = u_1 A + u_2 B$.
5. The verification $ver_{k_{pub}}(x, (r, s))$ follows from:

$$x_P \begin{cases} \equiv r \bmod q \implies \text{valid signature} \\ \not\equiv r \bmod q \implies \text{invalid signature} \end{cases}$$

Chapter 10 of *Understanding Cryptography* by Christof Paar and Jan Pelzl

Security of ECDSA

- It's difficult to find elliptic curves with good cryptographic properties
  - In practice, standard curves are used, from NIST or the Brainpool consortium

Bit lengths and security levels of ECDSA

q	Hash output (min)	Security levels
192	192	96
224	224	112
256	256	128
384	384	192
512	512	256

Lessons Learned

- Digital signatures provide message integrity, message authentication and non-repudiation.
- RSA is currently the most widely used digital signature algorithm.
- Competitors are the Digital Signature Standard (DSA) and the Elliptic Curve Digital Signature Standard (ECDSA).
- RSA verification can be done with short public keys *e*. Hence, in practice, RSA verification is usually faster than signing.
- DSA and ECDSA have shorter signatures than RSA
- In order to prevent certain attacks, RSA should be used with padding.
- The modulus of DSA and the RSA signature schemes should be at least 1024-bits long. For true long-term security, a modulus of length 3072 bits should be chosen. In contrast, ECDSA achieves the same security levels with bit lengths in the range 160–256 bits.

4