Cryptography 101: From Theory to Practice

Chapter 14 – Digital Signatures

Rolf Oppliger

May 13, 2022

Terms of Use

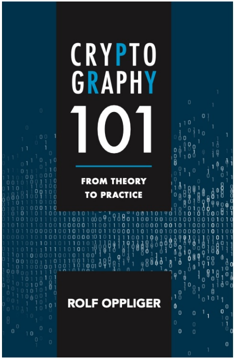
- This work is published with a CC BY-ND 4.0 license (🅭🅯⊜)
  - CC = Creative Commons (🅭)
  - BY = Attribution (🅯)
  - ND = No Derivatives (⊜)

whoami

rolf-oppliger.ch
rolf-oppliger.com

- Swiss National Cyber Security Centre NCSC (scientific employee)
- eSECURITY Technologies Rolf Oppliger (founder and owner)
- University of Zurich (adjunct professor)
- Artech House (author and series editor for information security and privacy)

Reference Book



© Artech House, 2021
ISBN 978-1-63081-846-3

https://books.esecurity.ch/crypto101.html

Challenge Me

Outline

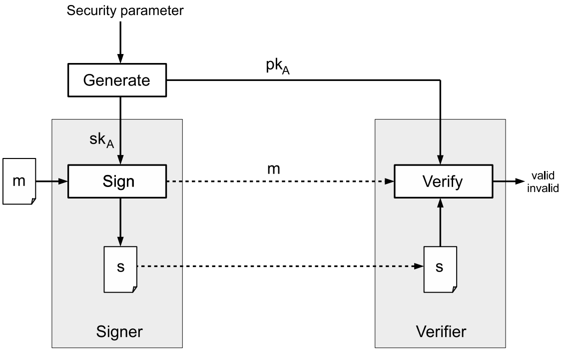
14. Digital Signatures

14. Digital Signatures

14. Digital Signatures

14.1 Introduction

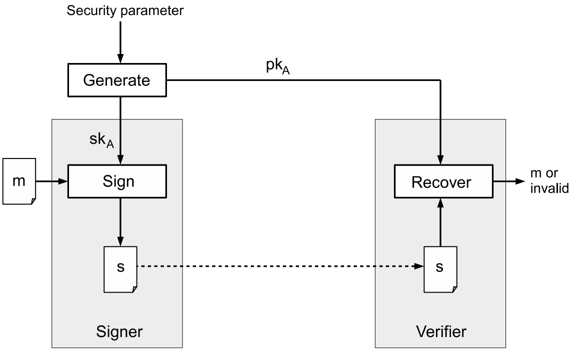
- DSS with appendix (cf. Definition 2.13)



14. Digital Signatures

14.1 Introduction

- DSS giving message recovery (cf. Definition 2.14)



14. Digital Signatures

14.1 Introduction

- A DSS must be correct and sound (secure)
  - It is **correct** if every valid signature is accepted
  - It is **sound** if no invalid siganture is accepted, i.e., it is computationally infeasible to forge a signature
- A proper **security definition** must specify
  1. The adversary's capabilities
  2. The task the adversary is required to solve in order to be successful (i.e., to break the security of the system)

14. Digital Signatures

14.1 Introduction

1. Classes of attacks (increasing power)
   - Key-only attacks
   - Known-message attacks (KMA)
   - Chosen-message attacks (CMA)
     - Nonadaptive
     - Adaptive
2. Types of forgery (decreasing difficulty)
   - Universal forgery
   - Selective forgery
   - Existential forgery

14. Digital Signatures

14.1 Introduction

- A **provably secure DSS** protects against existential forgery under an adaptive CMA
- This is the strongest notion of security
- The respective proof can be in the standard or random oracle model
- Most provably secure DSS follow the **hash-and-sign paradigm**
- A message is first hashed and then signed using a variant of a basic DSS, such as RSA, Rabin, or Elgamal

14. Digital Signatures

14.2 Digital Signature Systems

- RSA
- PSS and PSS-R
- Rabin
- Elgamal
- Schnorr
- DSA
- ECDSA
- Cramer-Shoup

14. Digital Signatures

14.2 Digital Signature Systems — RSA

Table 14.1
RSA DSS with Appendix

Domain parameters: —

Generate

(1^l)

$p, q \xleftarrow{r} \mathbb{P}_{l/2}$
$n = p \cdot q$
select $1 < e < \phi(n)$
with $gcd(e, \phi(n)) = 1$
compute $1 < d < \phi(n)$
with $de \equiv 1 \pmod{\phi(n)}$

$((n, e), d)$

Sign

(d, m)

$s = (h(m)^d) \bmod n$

(s)

Verify

$((n, e), m, s)$

$t = h(m)$
$t' = s^e \bmod n$
$b = (t = t')$

(b)

14. Digital Signatures

14.2 Digital Signature Systems — RSA

- Toy example
  - The Generate algorithm selects $p = 11$ and $q = 23$, computes $n = 11 \cdot 23 = 253$ and $\phi(253) = 10 \cdot 22 = 220$, selects $e = 3$, and computes $d = 147$ modulo 220 $[3 \cdot 147 = 441 \equiv 1 \pmod{220}]$
  - $(253, 3)$ is the public key, whereas 147 is the private key
  - To digitally sign a message m with $h(m) = 26$, the Sign algorithm computes $s = 26^{147} \bmod 253 = 104$
  - To verify the signature, the Verify algorithm computes $t = h(m)$ and $t' = \mathrm{RSA}_{253,3}(104) = 104^3 \bmod 253 = 26$, and returns *valid* (because $t = t' = 26$)

14. Digital Signatures

14.2 Digital Signature Systems — RSA

- Since $h(m)$ is typically much shorter than the modulus n, it is necessary to expand $h(m)$ to the bitlength of n
- This can be done by prepending zeros or using a distinct message expansion function (e.g., PKCS #1)
- Since PKCS #1 version 1.5, this function is

$$h_{\text{PKCS\#1}}(m) = \text{0x00 01 FF FF} \ldots \text{FF FF 00} \parallel h_{ID} \parallel h(m)$$

- PSS and PSS-R use a more sophisticated message expansion function (see below)

14. Digital Signatures

14.2 Digital Signature Systems — RSA

- If RSA is used as a DSS giving message recovery, then the Recover algorithm must first compute

$$m = \mathrm{RSA}_{n,e}(s) = s^e \bmod n$$

  and then decide whether m is a valid message
- Either the message is constructed in a natural language (that contains enough redundancy) or a redundancy scheme is used (e.g., $m \parallel m$ instead of m)

14. Digital Signatures

14.2 Digital Signature Systems — RSA

- The **security** of the RSA DSS depends on the properties of the RSA family of trapdoor permutations
- If one is able to factorize n, then one is also able to determine the private signing key sk and (universally) forge signatures
- Consequently, n must be so large that its factorization is computationally infeasible
- This means that $|n| \geq 2{,}048$ bits (or even 4,096 bits for high-value data)

14. Digital Signatures

14.2 Digital Signature Systems — RSA

- The multiplicative structure of the RSA function may be problematic in some application settings
- If m_1 and m_2 are two messages with signatures s_1 and s_2, then

$$s \equiv s_1 s_2 \equiv m_1^d m_2^d \equiv (m_1 m_2)^d \bmod n$$

  is a valid signature for $m = (m_1 m_2) \bmod n$
- Good practice in security engineering must take care of the multiplicative structure of the RSA function

14. Digital Signatures

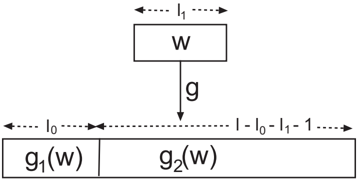
14.2 Digital Signature Systems — PSS and PSS-R

- Similar to OAEP in asymmetric encryption, PSS is a padding scheme that can be combined with a basic DSS, like RSA
- The resulting DSS is acronymed RSA-PSS
- RSA-PSS is provably secure in the radnom oracle model
- If PSS is combined with another DSS X, then the resulting DSS is acronymed X-PSS
- Examples include Rabin-PSS and Elgamal-PSS (not used in the field)

14. Digital Signatures

14.2 Digital Signature Systems — PSS and PSS-R

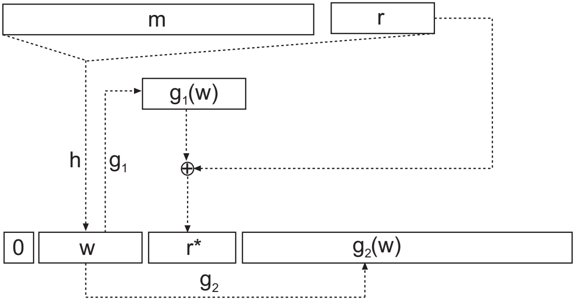
- The RSA key generation algorithm Generate prevails
- Additional parameters and functions
  - If l is the bitlength of n, then l_0 and l_1 are numbers between 1 and l (e.g., $l = 1,024$ and $l_0 = l_1 = 128$)
  - $h : \{0,1\}^* \rightarrow \{0,1\}^{l_1}$ is a "normal" hash function (compressor)
  - $g : \{0,1\}^{l_1} \rightarrow \{0,1\}^{l-l_1-1}$ is an XOF (generator)



14. Digital Signatures

14.2 Digital Signature Systems — PSS and PSS-R

- Preparation of argument for the RSA-PSS Sign algorithm



14. Digital Signatures

14.2 Digital Signature Systems — PSS and PSS-R

Table 14.2
RSA-PSS

Domain parameters: —

Sign

(d, m)

$r \xleftarrow{r} \{0,1\}^{l_0}$
$w = h(m \parallel r)$
$r^* = g_1(w) \oplus r$
$y = 0 \parallel w \parallel r^* \parallel g_2(w)$
$s = y^d \bmod n$

(s)

Verify

$((n,e), m, s)$

$y = s^e \bmod n$
break up y as $b \parallel w \parallel r^* \parallel \gamma$
$r = r^* \oplus g_1(w)$
$b = (b = 0 \wedge h(m \parallel r) = w \wedge g_2(w) = \gamma)$

(b)

14. Digital Signatures

14.2 Digital Signature Systems — PSS and PSS-R

- RSA-PSS is only slightly more expensive than basic RSA
- It was added in version 2.1 of PKCS #1
- The encoding method is referred to as EMSA-PSS
- EMSA-PSS stands for Encoding Method for Signature with Appendix
- PKCS #1 version 2.1 (EMSA-PSS) is widely used in the field
- It represents a (still) viable alternative for ECDSA

14. Digital Signatures

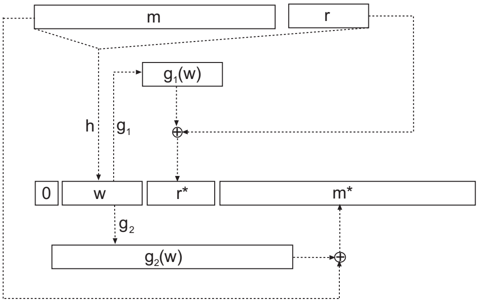
14.2 Digital Signature Systems — PSS and PSS-R

- PSS-R yields a DSS giving message recovery
- RSA-PSS-R uses the same parameters l, l_0, and l_1, and the same hash functions h and g (with g_1 and g_2)
- The messages to be signed have a maximum length $k = l - l_0 - l_1 - 1$
- Suggested choices are $l = 1,024$, $l_0 = l_1 = 128$, and $k = 767$
- This means that a 767-bit message m can be folded into the signature

14. Digital Signatures

14.2 Digital Signature Systems — PSS and PSS-R

- Preparation of argument for the RSA-PSS-R Sign algorithm



14. Digital Signatures

14.2 Digital Signature Systems — PSS and PSS-R

Table 14.3
RSA-PSS-R

Domain parameters: —

Sign	Recover
(d, m)	$((n, e), s)$
$r \xleftarrow{r} \{0,1\}^{l_0}$ $w = h(m \parallel r)$ $r^* = g_1(w) \oplus r$ $m^* = g_2(w) \oplus m$ $y = 0 \parallel w \parallel r^* \parallel m^*$ $s \equiv y^d \pmod{n}$	$y \equiv s^e \pmod{n}$ break up y as $b \parallel w \parallel r^* \parallel m^*$ $r = r^* \oplus g_1(w)$ $m = m^* \oplus g_2(w)$ if ($b = 0$ and $h(m \parallel r) = w$) then output m else output *invalid*
(s)	$(m \mid invalid)$

14. Digital Signatures

14.2 Digital Signature Systems — Rabin

- The Rabin public key cryptosystem yields another DSS
- It takes its security from the fact that computing square roots modulo n is computationally equivalent to factoring n
- Depending on its implementation (and the way the U-values are chosen), the Rabin DSS can be made provably secure in the random oracle model

14. Digital Signatures

14.2 Digital Signature Systems — Rabin

Table 14.4
Rabin DSS with Appendix (Simplified Version)

Domain parameters: —

Generate	Sign	Verify
(1^l)	$((p,q),m)$	$(n,m,(U,x))$
$p,q \xleftarrow{r} \mathbb{P}'_{l/2}$ $n = p \cdot q$	find U such that $h(m \parallel U)$ is a square modulo n find x that satisfies $x^2 \equiv h(m \parallel U) \pmod{n}$	$b = (x^2 \equiv h(m \parallel U) \pmod{n})$
$(n,(p,q))$	(U,x)	(b)

Again, $\mathbb{P}'_{l/2}$ refers to the set of all $l/2$-bit primes that are equivalent to 3 modulo 4 (so n is an l-bit Blum integer)

14. Digital Signatures

14.2 Digital Signature Systems — Elgamal

- The Elgamal public key cryptosystem yields another DSS with appendix
- A variant proposed by Kaisa Nyberg and Rainer R. Rueppel yields a DSS giving message recovery
- In contrast to RSA, the Elgamal DSS uses different algorithms and signatures that are twice as long as the modulus
- It is therefore not widely used in the field
- It is defined in a cyclic group in which the DLP is computationally intractable, such as $\mathbb{Z}_p^*$ (original proposal)

14. Digital Signatures

14.2 Digital Signature Systems — Elgamal

Table 14.5
Elgamal DSS with Appendix

Domain parameters: p, g

Generate	Sign	Verify
$(-)$	(x, m)	$(y, m, (s_1, s_2))$
$x \xleftarrow{r} \{2, \ldots, p-2\}$ $y = g^x \bmod p$	$r \xleftarrow{r} \{1, \ldots, p-2\}$ with $gcd(r, p-1) = 1$ $s_1 = g^r \bmod p$ $s_2 = (r^{-1}(h(m) - xs_1)) \bmod (p-1)$	verify $0 < s_1 < p$ verify $0 < s_2 < p-1$ $b = (g^{h(m)} \equiv y^{s_1} s_1^{s_2} \pmod{p})$
(x, y)	(s_1, s_2)	(b)

14. Digital Signatures

14.2 Digital Signature Systems — Elgamal

- The verification is correct, because

$$\begin{aligned} y^{s_1} s_1^{s_2} &\equiv g^{xs_1} g^{rr^{-1}(h(m)-xs_1)} \pmod{p} \\ &\equiv g^{xs_1} g^{h(m)-xs_1} \pmod{p} \\ &\equiv g^{xs_1} g^{-xs_1} g^{h(m)} \pmod{p} \\ &\equiv g^{h(m)} \pmod{p} \end{aligned}$$

14. Digital Signatures

14.2 Digital Signature Systems — Elgamal

- Toy example
  - For $p = 17$ ($\mathbb{Z}_{17}^*$) and $g = 7$, the Generate algorithm selects $x = 6$ and computes $y = 7^6 \bmod 17 = 9$
  - 9 is the public key and 6 is the private key
  - To digitally sign m with $h(m) = 6$, the Sign algorithm selects $r = 3$ (with $r^{-1} \equiv 3^{-1} \pmod{16} = 11$) and computes

$$\begin{aligned} s_1 &= 7^3 \bmod 17 = 343 \bmod 17 = 3 \\ s_2 &= (11(6 - 6 \cdot 3)) \bmod 16 = -132 \bmod 16 = 12 \end{aligned}$$

  - The signature is (3,12)
  - The Verify algorithm must verify $0 < 3 < 17$, $0 < 12 < 16$, and $7^6 \equiv 9^3 \cdot 3^{12} \pmod{17}$, which is 9 in either case

14. Digital Signatures

14.2 Digital Signature Systems — Elgamal

- The security of the Elgamal public key cryptosystem is based on the assumed intractability of the DLP in a cyclic group
- In $\mathbb{Z}_p^*$, p must be at least 2,048 bits
- Furthermore, one must select p so that efficient algorithms to compute discrete logarithms do not work
- For example, $p - 1$ must not have only small prime factors (otherwise, the Pohlig-Hellman algorithm can be applied)
- Furthermore, h must be a cryptographic hash function and r must be unique and unpredictable

14. Digital Signatures

14.2 Digital Signature Systems — Schnorr

- Claus-Peter Schnorr proposed the use of a q-order subgroup G of $\mathbb{Z}_p^*$ with $q \mid p-1$ (Schnorr group)
- For example, for $p = 23$ and $q = (23-1)/2 = 11$, $g = 2$ is a generator of the Schnorr group $\{1, 2, 3, 4, 6, 8, 9, 12, 13, 16, 18\}$ with 11 elements ($g = 4$ is another generator)
- All computations are done in the Schnorr group
- Originally, $|p|=$ 1,024 bits and $|q|$=160 bits
- The Schnorr DSS is more efficient and the signatures are shorter ($2 \cdot 160 = 320$ instead of $2 \cdot 2,048 = 4,096$ bits)

14. Digital Signatures

14.2 Digital Signature Systems — Schnorr

- Unlike Elgamal, the Schnorr DSS relies on the DLP in the q-order subgroup of $\mathbb{Z}_p^*$
- This problem can only be solved with a generic algorithm
- Such an algorithm has a running time that is of the order of the square root of q
- For $|q|$=160 bits, the subgroup has order 2^{160} and the running time is of the order of $\sqrt{2^{160}} = 2^{160/2} = 2^{80}$
- The Schnorr DSS is with appendix, but it can be turned into a DSS giving message recovery (not addressed here)

14. Digital Signatures

14.2 Digital Signature Systems — Schnorr

Table 14.6
Schnorr DSS

Domain parameters: p, q, g

Generate	Sign	Verify
$(-)$	(x, m)	$(y, m, (s_1, s_2))$
$x \xleftarrow{r} \mathbb{Z}_q^*$ $y = g^x \bmod p$	$r \xleftarrow{r} \mathbb{Z}_q^*$ $r' = g^r \bmod p$ $s_1 = h(r' \parallel m)$ $s_2 = (r + xs_1) \bmod q$	$u = (g^{s_2} y^{-s_1}) \bmod p$ $v = h(u \parallel m)$ $b = (v = s_1)$
(x, y)	(s_1, s_2)	(b)

14. Digital Signatures

14.2 Digital Signature Systems — Schnorr

- The verification is correct, because $v = s_1$ suggests that $h(u \parallel m) = h(r' \parallel m)$ and hence $u = r'$
- This equation is true, because

$$\begin{aligned} u &= (g^{s_2} y^{-s_1}) \bmod p \\ &= (g^{r+xs_1} g^{-xs_1}) \bmod p \\ &= (g^r g^{xs_1} g^{-xs_1}) \bmod p \\ &= g^r \bmod p \\ &= r' \end{aligned}$$

14. Digital Signatures

14.2 Digital Signature Systems — Schnorr

- Toy example
  - For $p = 23$, $q = 11$, and $g = 2$ (see above), the Generate algorithm selects $x = 5$ and computes $y = 2^5 \bmod 23 = 9$
  - 9 is the public key and 5 is the private key
  - To digitally sign m, the Sign algorithm selects $r = 7$ and computes $r' = 2^7 \bmod 23 = 13$
  - If $h(r' \parallel m) = 4$, then $s_1 = 4$ and $s_2 = (7 + 5 \cdot 4) \mod 11 = 5$
  - The signature is (4,5)
  - The Verify algorithm computes $u = (2^5 \cdot 9^{-4}) \bmod 23 = 13$ and $v = h(13 \parallel m) = 4$, and accepts the signature (because $v = s_1 = 4$)

14. Digital Signatures

14.2 Digital Signature Systems — DSA

- Based on the DSS of Elgamal and Schnorr, NIST developed the digital signature algorithm (DSA) and digital signature standard (DSS) in FIPS PUB 186
- Since its publication in 1994, FIPS PUB 186 has been subject to 4 major revisions in 1998, 2000, 2009, and 2013
- Originally, p had a variable bitlength ($512 + 64t$ bits for $t \in \{0, \ldots, 8\}$), and q was fixed to 160 bits
- More recent revisions of FIPS PUB 186 support longer bitlengths for p and q, as well as RSA and ECDSA

14. Digital Signatures

14.2 Digital Signature Systems — DSA

Table 14.7
DSA

Domain parameters: p, q, g

Generate	Sign	Verify
$(-)$	(x, m)	$(y, m, (s_1, s_2))$
$x \xleftarrow{r} \mathbb{Z}_q^*$ $y = g^x \bmod p$	$r \xleftarrow{r} \mathbb{Z}_q^*$ $s_1 = (g^r \bmod p) \bmod q$ $s_2 = r^{-1}(h(m) + xs_1) \bmod q$	verify $0 < s_1, s_2 < q$ $w = s_2^{-1} \bmod q$ $u_1 = (h(m)w) \bmod q$ $u_2 = (s_1 w) \bmod q$ $v = (g^{u_1} y^{u_2} \bmod p) \bmod q$ $b = (v = s_1)$
(x, y)	(s_1, s_2)	(b)

14. Digital Signatures

14.2 Digital Signature Systems — DSA

- Toy example
  - For $p = 23$, $q = 11$, and $g = 4$, the Generate algorithm selects $x = 3$ and computes $y = 4^3 \bmod 23 = 18$
  - 18 is the public key and 3 is the private key
  - To digitally sign m with $h(m) = 6$, the Sign algorithm selects $r = 7$, computes $s_1 = (4^7 \bmod 23) \bmod 11 = 8$, determines $r^{-1} = 7^{-1} \bmod 11 = 8$, and computes $s_2 = 8(6 + 8 \cdot 3) \bmod 11 = 9$
  - The signature is (8,9)
  - The Verify algorithm verifies that $0 < 8, 9 < 11$, computes $w = 9^{-1} \bmod 11 = 5$, $u_1 = (6 \cdot 5) \bmod 11 = 8$, $u_2 = (8 \cdot 5) \bmod 11 = 7$, and $v = (4^8 18^7 \bmod 23) \bmod 11 = 8$, and returns *valid* (because $v = s_1 = 8$)

14. Digital Signatures

14.2 Digital Signature Systems — ECDSA

- ECDSA refers to the elliptic curve variant of DSA
- Instead of working in a q-order subgroup of $\mathbb{Z}_p^*$, it works in a group of points on an elliptic curve over a finite field $\mathbb{F}_q$, i.e., $E(\mathbb{F}_q)$, where q is an odd prime or a power of 2
- Today, ECDSA is most widely deployed in the field
- It has been adopted in many standards, including ANSI X9.62, NIST FIPS 186, ISO/IEC 14888, IEEE 1363-2000, and the standards for efficient cryptography (SEC) 1 and 2

14. Digital Signatures

14.2 Digital Signature Systems — ECDSA

Table 14.8
ECDSA

Domain parameters: *Curve*, G, n

Generate

$(-)$

$$d \xleftarrow{r} \mathbb{Z}_n^*$$
$$Q = dG$$

(d, Q)

Sign

(d, m)

$$z = h(m)\,|_{len(n)}$$
$$r \xleftarrow{r} \mathbb{Z}_n^*$$
$$(x_1, y_1) = rG$$
$$s_1 = x_1 \bmod n$$
$$s_2 = r^{-1}(z + s_1 d) \bmod n$$

(s_1, s_2)

Verify

$(Q, m, (s_1, s_2))$

verify legitimacy of Q
verify $0 < s_1, s_2 < n$
$$z = h(m)\,|_{len(n)}$$
$$w = s_2^{-1} \bmod n$$
$$u_1 = (zw) \bmod n$$
$$u_2 = (s_1 w) \bmod n$$
$$(x_1, y_1) = u_1 G + u_2 Q$$
$$b = ((x_1, y_1) \neq \mathcal{O}) \wedge (s_1 = x_1)$$

(b)

14. Digital Signatures

14.2 Digital Signature Systems — ECDSA

- Toy example ($Curve = E(\mathbb{Z}_{23})$, $G = (3,10)$, $n = 28$)
  - The Generate algorithm randomly selects $d = 3$ and computes $Q = 3 \cdot G = 3 \cdot (3,10) = (19,5)$
  - $(19,5)$ is the public key and 3 is the private key
  - To digitally sign m, whose leftmost $len(n)$ bits of $h(m)$ refers to $z = 5$, the Sign algorithm randomly selects $r = 11$, and computes $11 \cdot G = (18,20)$, $s_1 = 18$, and $s_2 = 23(5 + 18 \cdot 3) \bmod 28 = 13$
  - The signature is (18,13)
  - The Verify algorithm verifies $0 < 18, 13 < 28$, computes $w = 13$, $u_1 = (5 \cdot 13) \bmod 28 = 9$, $u_2 = (18 \cdot 13) \bmod 28 = 10$, and $9 \cdot G + 10 \cdot Q = (18,20)$
  - Because this point is $\neq \mathcal{O}$ and its x-coordinate equals s_1, the algorithm returns *valid*

14. Digital Signatures

14.2 Digital Signature Systems — ECDSA

- Unlike the DSA, the ECDSA is provably secure, i.e., it protects against existential forgery under an adaptive CMA
- The proof is in the random oracle model
- Like Elgamal and all variants, r must be unique and unpredictable
- Dan Boneh, Ben Lynn, and Hovav Shacham proposed a variant of ECDSA based on bilinear maps (BLS)
- Because such signatures are very short (160 bits) and can be aggregated, they are widely used in blockchain applications

14. Digital Signatures

14.2 Digital Signature Systems — Cramer-Shoup

- All practical DSS addressed so far have either no security proof or "only" a security proof in the random oracle model
- This is different with the Cramer-Shoup DSS
- This DSS is practical and can be proven secure in the standard model under the strong RSA assumption
- There is also a variant that can be proven secure in the random oracle model under the standard RSA assumption (not addressed here)

14. Digital Signatures

14.2 Digital Signature Systems — Cramer-Shoup

Table 14.9

Cramer-Shoup DSS

Domain parameters: l, l' (e.g., $l = 160$ and $l' = 512$)

Generate	Sign	Verify
$(-)$	(sk, m)	(pk, m, s)
$p, q \xleftarrow{r} \mathbb{P}^*_{l'}$ $n = pq$ $f, x \xleftarrow{r} QR_n$ $e' \xleftarrow{r} \mathbb{P}_{l+1}$ $pk = (n, f, x, e')$ $sk = (p, q)$	$e \xleftarrow{r} \mathbb{P}_{l+1}$ with $e \neq e'$ $y' \xleftarrow{r} QR_n$ solve $(y')^{e'} = x' f^{h(m)}$ for x' solve $y^e = x f^{h(x')}$ for y $s = (e, y, y')$	verify $e \neq e'$ verify e is odd verify $len(e) = l + 1$ compute $x' = (y')^{e'} f^{-h(m)}$ $b = (x = y^e f^{-h(x')})$
(pk, sk)	(s)	(b)

$\mathbb{P}^*_{l'}$ refers to the set of all safe primes with bitlength l', whereas $\mathbb{P}_{l+1}$ refers to the set of all primes with bitlength $l + 1$

14. Digital Signatures

14.3 Identity-Based Signatures

- In the early 1980s, Adi Shamir came up with the idea of identity-based cryptography and proposed a respective DSS
  - A trusted authority chooses an RSA modulus n (with prime factors p and q), a large integer e with $gcd(e, \phi(n))$, and a one-way function f as domain parameters
  - For every user, it derives a public key pk from the user's identity, and computes the respective private key sk as the e-th root of pk modulo n, i.e., $sk^e \equiv pk \pmod{n}$
  - It can do so, only because it knows the prime factorization of n

14. Digital Signatures

14.3 Identity-Based Signatures

- To sign message $m \in \mathbb{Z}_n$, the user selects $r \in_R \mathbb{Z}_n$ and computes $t = r^e \bmod n$ and $s = (sk \cdot r^{f(t,m)}) \bmod n$
- The signature is (s, t)
- It is valid, if $s^e \equiv pk \cdot t^{f(t,m)} \pmod{n}$ holds

$$\begin{aligned} s^e &\equiv (sk \cdot r^{f(t,m)})^e \pmod{n} \\ &\equiv sk^e r^{ef(t,m)} \pmod{n} \\ &\equiv pk \cdot t^{f(t,m)} \pmod{n} \end{aligned}$$

14. Digital Signatures

14.3 Identity-Based Signatures

- Shamir's identity-based DSS has fueled a lot of research and development in identity-based cryptography
- Many other identity-based DSS have been proposed (but only a few IBE systems)
- Main disadvantages
  - Unique naming scheme is needed
  - Trusted authority is needed (to issue public key pairs)
  - Key revocation is still needed

14. Digital Signatures

14.4 One-Time Signatures

- In a one-time signature system a public key pair can be used to sign a single message
- If the pair is reused, then it may become feasible to forge a signature
- The advantages are related to simplicity and efficiency
- The disadvantages are related to the size of the verification key(s) and the overhead related to key management
- One-time signatures are often combined with techniques to efficiently authenticate public keys, such as Merkle trees

14. Digital Signatures

14.4 One-Time Signatures

- Historically, the first one-time signature system was proposed by Michael O. Rabin in 1978
- The system employed a symmetric encryption system and was too inefficient to be used in practice
- In 1979, Leslie Lamport proposed a system that is efficient because it only employs a one-way function f
- If combined with techniques to efficiently authenticate public verification keys (e.g., Merkle trees), the resulting one-time signature system is practical

14. Digital Signatures

14.4 One-Time Signatures

- Let f be a one-way function and m a message to be signed
- Let the bitlength of m be at most n, e.g., 128 or 160 bits (otherwise m is first hashed)
- The signatory must have a private key that consists of n pairs of randomly chosen preimages for f:

$$[u_{10}, u_{11}], [u_{20}, u_{21}], \ldots, [u_{n0}, u_{n1}]$$

- Each u_{ij} ($i = 1, \ldots, n$ and $j = 0, 1$) may be an n-bit string
- The $2n$ arguments may be generated with a PRG

14. Digital Signatures

14.4 One-Time Signatures

- The respective public key consists of the $2n$ images $f(u_{ij})$:

$$[f(u_{10}), f(u_{11})], [f(u_{20}), f(u_{21})], \ldots, [f(u_{n0}), f(u_{n1})]$$

- The $2n$ images $f(u_{ij})$ are hashed to a single value p that represents the public key:

$$p = h(f(u_{10}), f(u_{11}), f(u_{20}), f(u_{21}), \ldots, f(u_{n0}), f(u_{n1}))$$

- Complementary techniques to efficiently authenticate verification keys are needed for multiple signatures

14. Digital Signatures

14.4 One-Time Signatures

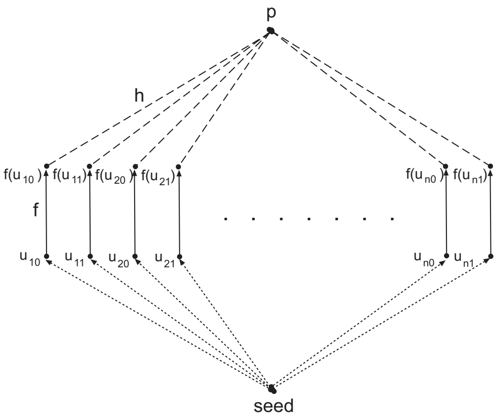
- To sign message m, each bit m_i $(i = 1, \ldots, n)$ must be individually signed using the preimage pair $[u_{i0}, u_{i1}]$
  - If $m_i = 0$, the signature comprises u_{i0}
  - If $m_i = 1$, the signature comprises u_{i1}
- The signature s for m comprises all such values

$$s = [u_{1m_1}, u_{2m_2}, \ldots, u_{nm_n}]$$

- It can be verified by computing all images $f(u_{ij})$, hashing all values to p', and comparing p' with p (it is valid if $p' = p$)

14. Digital Signatures

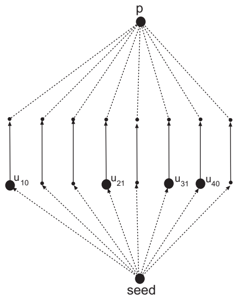
14.4 One-Time Signatures



14. Digital Signatures

14.4 One-Time Signatures

- Exemplary one-time signature for message $m = 0110$
- Message bit m_1 is signed with u_{10}, m_2 with u_{21}, m_3 with u_{31}, and m_4 with u_{40}



14. Digital Signatures

14.4 One-Time Signatures

- There are several possibilities to generalize and improve the Lamport one-time DSS
- Some improvements are due to Merkle
- Other improvements have been proposed recently to make one-time signatures suitable for PQC (e.g., SPHINCS+)
- The Lamport one-time DSS and some variants are used in many cryptographic applications (e.g., anonymous offline digital cash)

14. Digital Signatures

14.5 Variants

- Blind signatures
- Undeniable signatures
- Fail-stop signatures
- Group signatures (ring signatures)
- ...

14. Digital Signatures

14.6 Final Remarks

- Digital signatures provide the digital analog of handwritten signatures
- They are necessary to provide nonrepudiation services
- Many countries and communities have legislation
  - U.S. Electronic Signatures in Global and National Commerce Act, commonly referred to as ESIGN (2000)
  - European Electronic Identification and Trust Services Regulation, commonly referred to as eIDAS (2014)
- This also applies to Switzerland (OFCOM)

14. Digital Signatures

14.6 Final Remarks

- But the laws on electronic or digital signatures have not yet been disputed in court
- It is therefore not clear what their legal status is
- Signatures always depend on many layers of hardware and software
- On each of these layers (including the user on top of them), many things can go wrong
- The mathematical precision of digital signatures in theory is blurred in practice

Questions and Answers

Thank you for your attention