Cryptography
Lecture 7

Dr. Panagiotis Rizomiliotis

Agenda

- Digital Signatures
- Elliptic curve cryptography

DIGITAL SIGNATURES

Digital Signature

- Schemes used to provide
  - authentication,
  - integrity and
  - non-repudiation services (difficult, strong bidding, legal force)
- Asymmetric analogue of MACs
- Consist of three algorithms:
  - KeyGen(λ)→(sk,vk)
  - Sign(sk,m)→σ
  - Verify(vk,σ)→{0,1}

DSS VS MAC

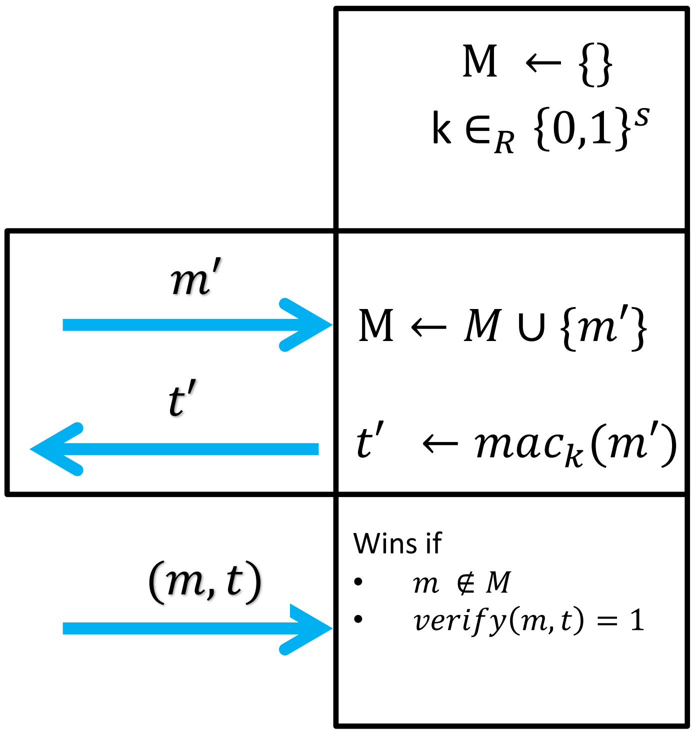
- $Gen(1^n) \;\rightarrow\; (sk, vk)$
- $Sign_{sk}(m) \;\rightarrow\; sig$
- $Ver_{vk}(m, sig) \rightarrow \; \{0,1\}$

- $Gen(1^n) \;\rightarrow\; k$
- $mac_k(m) \;\rightarrow\; t$
- $\mathrm{v}er_k(m, t) \rightarrow \; \{0,1\}$

Security definition

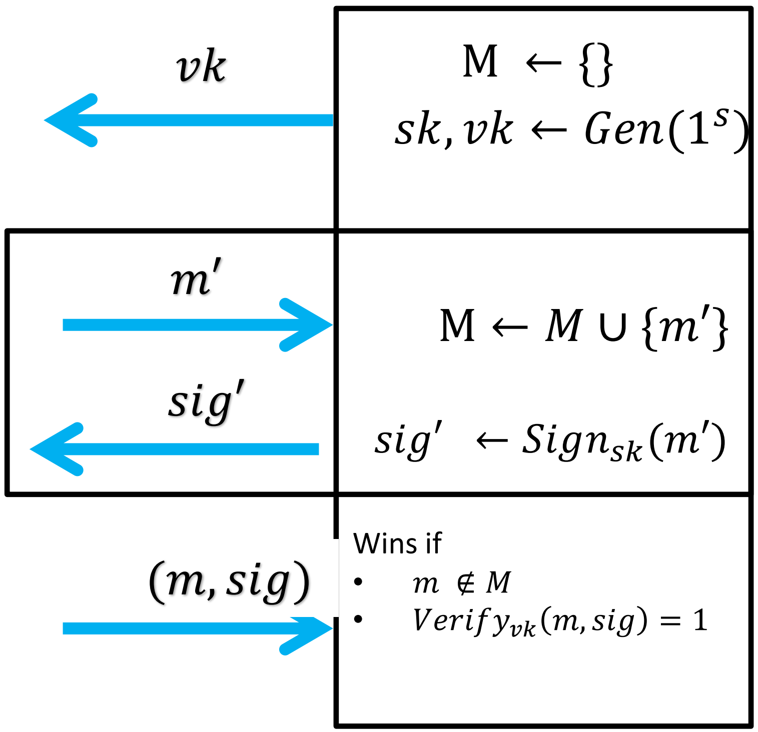
- Similar to MAC:
  1. Many pairs (m1, σ1),(m2, σ2), . . . produced by Sign (chosen)
  2. Produce an new one (m,σ) that verifies under the key *vk.*
- The formal security notion is called **Strong Unforgeability under Chosen Message Attack (SUF-CMA)**

Mac forgery game



Repeat as many times as the adversary wants

Signature forgery game



Repeat as many times as the adversary wants

Definition of signature scheme

- Correctness:
  - $\Pr\left[Ver_{vk}\left(m, Sign_{sk}(m)\right) = 1 \mid \ (sk, vk) \leftarrow Gen(1^s)\right] = 1$
- Unforgeability
  - For all PPT adversary A, there exists negligible function μ,
    - $\Pr[\ A \ \ wins\ the\ signature\ forgery\ game] \leq \mu(n)$

Relation between macs and signatures

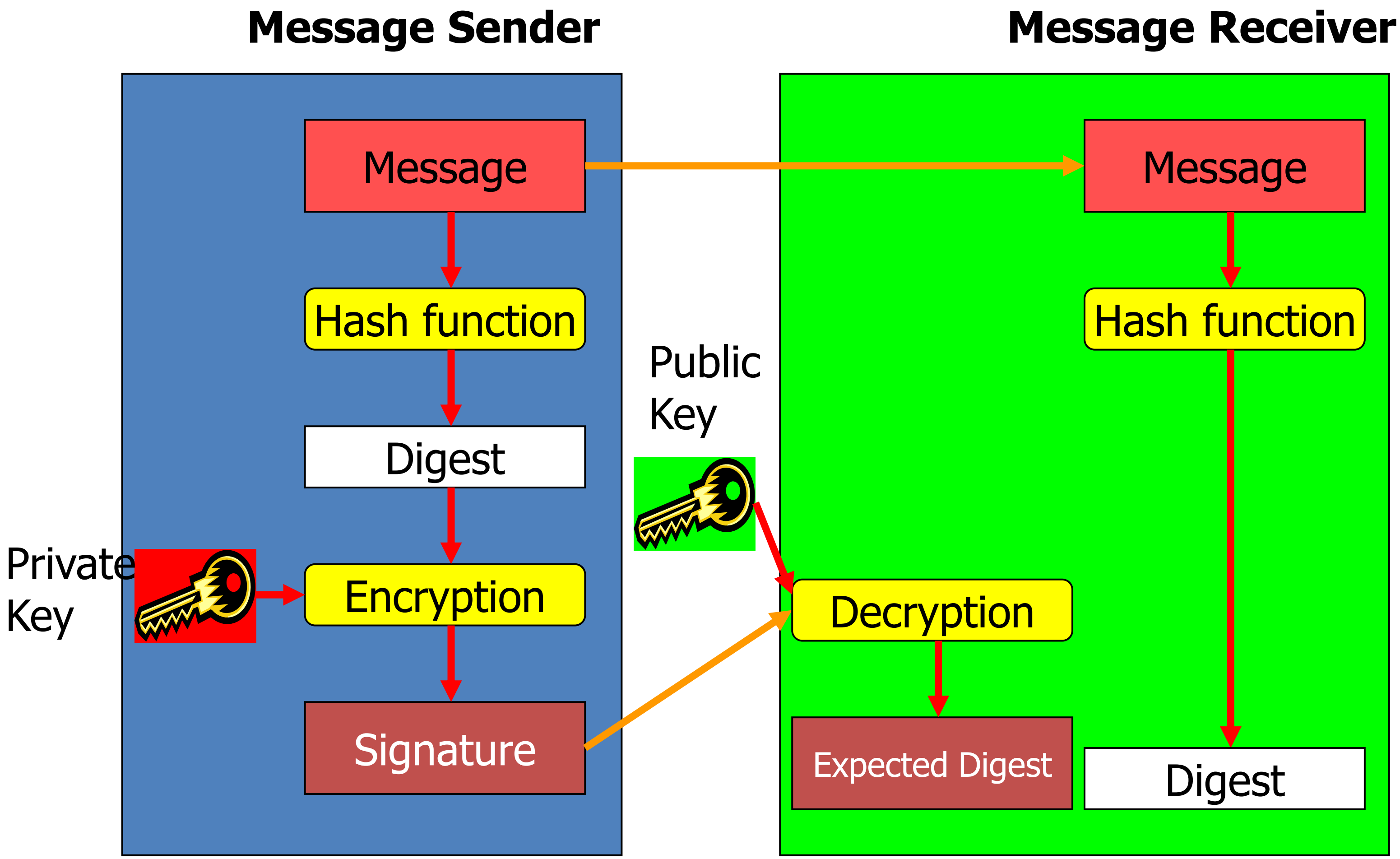
- Every signature scheme is a message authentication code.

- A mac scheme is not necessarily a signature.
  - Without the key, it may be impossible to verify a mac.

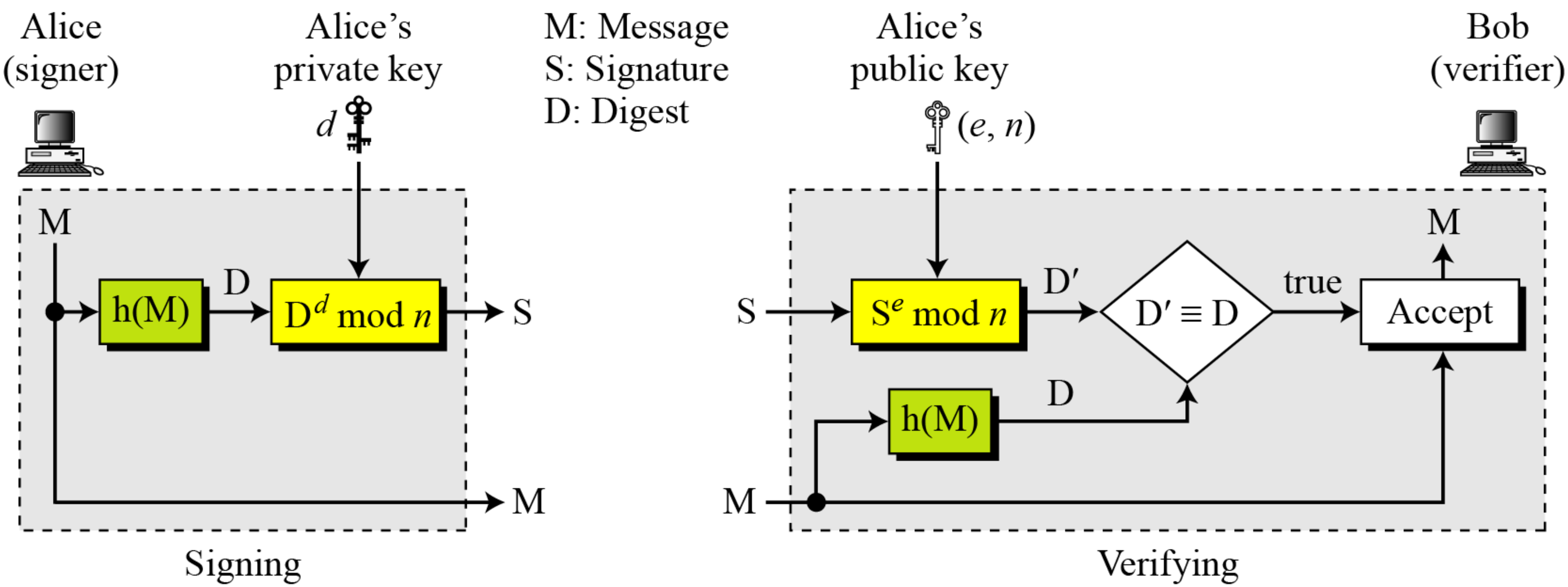
Security (cont.)

- common pitfall:
  - we assume that a signature σ must bind a message m and a verification key vk
  - the SUF-CMA security definition does not imply this!!!
  - it only refers to security under **a single** key pair (sk, vk)!!
- Duplicate Signature Key Selection (DSKS) attacks!

Digital Signatures



RSA + hash



Alice
(signer)
Alice's
private key
d
M: Message
S: Signature
D: Digest
Alice's
public key
(e, n)
Bob
(verifier)
M
h(M)
D
$D^d \bmod n$
S
M
Signing
S
$S^e \bmod n$
D′
D′ ≡ D
true
Accept
M
h(M)
D
M
Verifying

RSA Std solutions

- RSA-PKCS# 1 v1.5
  – Has no security proof,
  – Nor any advantages over other RSA
  – it is widely deployed.
  – Not propose be used beyond legacy systems.
- RSA-PSS
  – UF-CMA secure in the random oracle model
  – It is used in a number of places including e-passports.
- RSA-FDH
  – The RSA-FDH scheme hashes the message to the group Z/NZ and then applies the RSA function to the output.
  – The scheme has strong provable security guarantees
  – Difficult to defining a suitably strong hash function with codomain the group Z=NZ.
  – The scheme is not practically deployable.

Std solutions

- ISO 9796-2 RSA Based Mechanisms
  - 3 different RSA signature padding schemes called Digital Signature 1,Digital Signature 2 and Digital Signature 3 (DS1, DS2 and DS3).
  - Variant DS1 essentially RSA encrypts a padded version of the message along with a hash of the message. This variant should no longer be considered secure.
  - Variant DS2 is a standardized version of RSA-PSS, but in a variant which allows partial message recovery.
  - Variant DS3 is defined by taking DS2 and reducing the randomisation parameter to length zero. Not to use for future applications

From ElGamal to DSA

- The Digital Signature Algorithm (DSA) is a modification of ElGamal digital signature scheme.
- It was proposed in August 1991 and adopted in December 1994 by the National Institute of Standards and Technology.
- Digital Signature Standard (DSS)
  - ✓ Computation of DSS signatures is faster than computation of RSA signatures when using the same p.
  - ✓ DSS signatures are smaller than ElGamal signatures because q is smaller than p.

Digital Signature Algorithm (DSA)

Also known as Digital Signature Standard (DSS)

Key generation

- Select two prime numbers (p,q) such that $q \mid (p-1)$
- Early standard recommended p to be between 512 and 1024 bits, and q to be 160 bits
- Current recommendation for length: (1024,160), (2048,224), (2048,256), and (3072,256).
  - The size of q must resist exhaustive search
  - The size of p must resist discrete log
- Choose g to be an element in Z_p^* with order q
  - Let α be a generator of Z_p^*, and set $g = \alpha^{(p-1)/q} \bmod p$
- Select $1 \leq x \leq q-1$; Compute $y = g^x \bmod p$

Public key: (p, q, g, y)

Private key: x

DSA

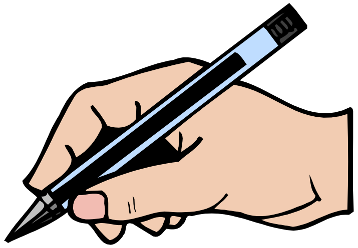
Signing message M:

- Select a random integer k, $0 < k < q$
- Compute

  $$r = (g^k \bmod p) \bmod q$$

  $$s = k^{-1} (h(M) + xr) \bmod q$$

- Signature: (r, s)
  - Signature consists of two 160-bit numbers, when q is 160 bit



DSA

Signature: (r, s)

$r = (g^k \bmod p) \bmod q$

$s = k^{-1} (h(M) + xr) \bmod q$

Verification

- Verify $0 < r < q$ and $0 < s < q$, if not, invalid
- Compute

  $u_1 = h(M)s^{-1} \bmod q$,

  $u_2 = rs^{-1} \bmod q$
- Valid iff $r = (g^{u_1} y^{u_2} \bmod p) \bmod q$

  $g^{u_1} y^{u_2} = g^{h(M)s^{-1}} g^{xr\, s^{-1}}$

  $= g^{(h(M)+xr)s^{-1}} = g^k \pmod p$

Schnorr signature scheme

- Requirement: Group G, $|G| = q$, generator g, random oracle H

- $Gen(1^s)$
  - $sk \in_R \; G$
  - $vk \leftarrow g^{sk}$

- $Verify_{vk}(m, sig)$
  - $(a, s) \leftarrow sig$
  - $\mathrm{u} \leftarrow g^s \cdot vk^{-a}$
  - Output $H(u, m) = a$

- $Sign_{sk}(m)$
  - $b \in_R Z_{|G|}$
  - $u \leftarrow g^b$
  - $a \leftarrow H(u, m)$
  - $s \leftarrow a \cdot sk + b \; (mod \;\; q)$
  - Output (a, s)

EdDSA

- Introduced in 2011 by Bernstein, Duif, Lange, Schwabe, and Yang in the paper "High-speed high-security signatures"
- Modified version of Schnorr Signatures
- Based on twisted Edwards curves
- Most known the Ed25519
  - using SHA-512 (SHA-2) and Curve25519
  - TLS 1.3, SSH, Tor, ZCash, Signal protocol, WhatsApp
- Standards
  - IETF, RFC 8032
  - NIST, as part of FIPS 186–5 (2019)

Performance

Algorithm	Public Key	Signature	Sign/s
ED25519	32B	64B	~ 26,000
RSA-2048	0.3kB	0.3kB	~1,500

Std solutions

- PV Signatures
  - ISO 14888-3
  - A variant of DSA signatures (exactly the same signing equation as for DSA)
  - Due to Pointcheval and Vaudeney
  - The PV signature scheme can be shown to be provably secure in the random oracle model
  - PV signatures suffer from issues related to poor randomness in the ephemeral secret key.
- (EC)Schnorr
  - Like (EC)DSA signatures
  - Schnorr signatures can be proved UF-CMA secure in the random oracle model [280].
  - Also a proof in the generic group model
  - Signature size can be made shorter than that of DSA.
  - Schnorr signatures are to be preferred over DSA style signatures for future applications.
  - Defences proposed for (EC)DSA signatures should also be applied to Schnorr signatures

Std solutions

- (EC)DSA
  - Widely standardized
    - German DSA (GDSA),
    - Korean DSA (KDSA)
    - Russian DSA (RDSA) [133,162].
  - All (EC)DSA variants (bar KDSA) have weak provable security guarantees
  - The KDSA is suitable for future use.

More on Signatures

- *Blind Signatures*

Sometimes we have a document that we want to get signed without revealing the contents of the document to the signer.

- *Group Signatures*

Protect privacy. Part of a group. Not the same secret key. A manager can reveal identity

- *Ring Signatures*

Protect privacy. Part of a group. Not the same secret key. The cryptocurrency Monero uses ring signatures to provide anonymity

- *Time Stamped Signatures*

Sometimes a signed document needs to be time stamped to prevent it from being replayed by an adversary. This is called time-stamped digital signature scheme.

- *Proxy Signatures*

Delegate signature to a server.

Blind Signature Schemes

- A wants B's signature on a message m, but doesn't want B to know the message m or the signature
- Applications: electronic cash
  - Goal: anonymous spending
  - The bank signs a bank note, but A doesn't want B to know the note, as then B can associate the spending of B with A's identity

Chaum’s Bind Signature Protocol Based on RSA

- Setup:
  - B has public key (n,e) and private key d
  - A has m
- Actions:
  - (blinding) A picks random $k \in Z_n - \{0\}$ computes $m' = mk^e \bmod n$ and sends to B
  - (signing) B computes $s' = (m')^d \bmod n$ and sends to A
  - (unblinding) A computes $s = s'k^{-1} \bmod n$, which is B’s signature on m

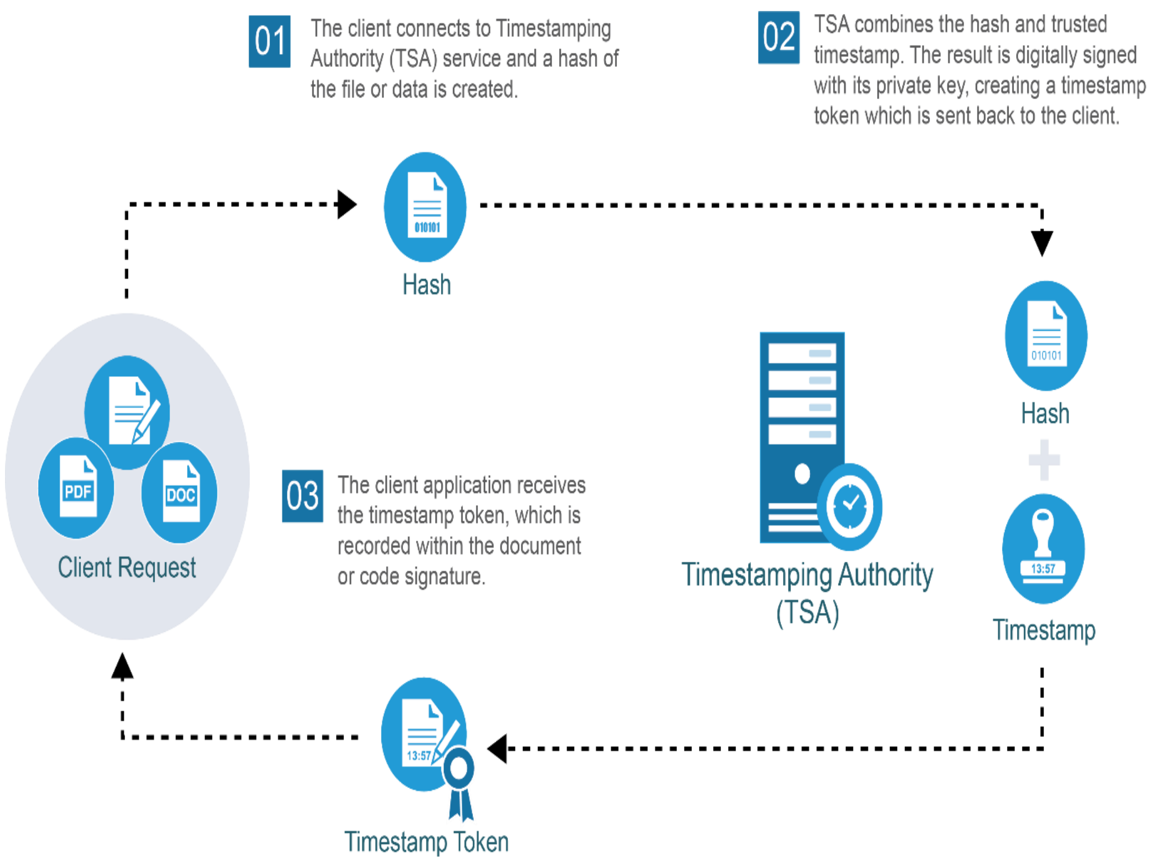
Timestamping

- Timestamping is very valuable
- Trusted Timestamp
  - timestamps are generated by a trusted third party using secure FIPS-compliant hardware
  - high level of certainty that the date on the timestamp is accurate and hasn't been tampered with
- RFC 3161 outlines the requirements a third party must meet in order to operate as a Timestamping Authority (TSA)

Timestamping

1. The client application creates a hashed value (as a unique identifier of the data or file that needs to be timestamped) and sends it to the TSA.
2. From now on, any change (even by a single bit of information) in the original file will require communication of changes with the TSA server.
3. The TSA combines the hash and other information, including the authoritative time. The result is digitally signed with the TSA's private key, creating a timestamp token which is sent back to the client. The timestamp token contains the information the client application will need to verify the timestamp later.
4. The timestamp token is received by the client application and recorded within the document or code signature.

Timestamping



One-Time Digital Signatures

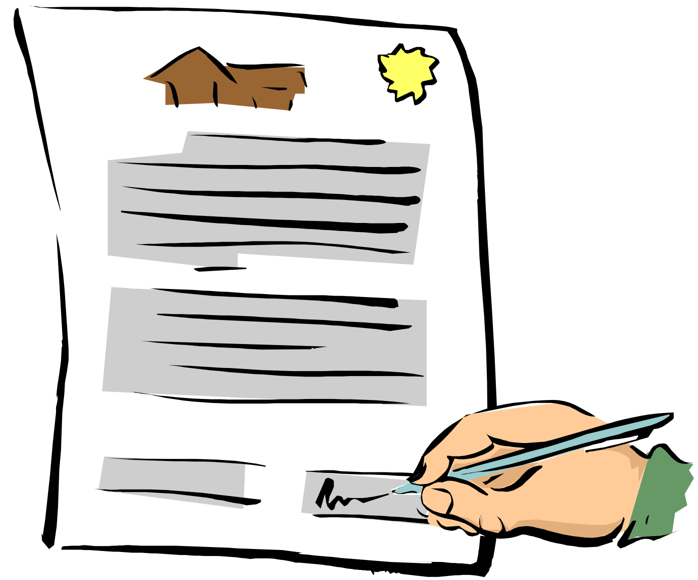
- One-time digital signatures: digital schemes used to sign, at most one message; otherwise signature can be forged.
- A new public key is required for each signed message.
- Advantage: signature generation and verification are very efficient and is useful for devices with low computation power.

- Used by the hash-based signature scheme SPHINCS+
  - It is an “alternate candidate” in the NIST PQC process for selecting post-quantum secure schemes

Lamport One-time Signature

To sign one bit:

- Choose as secret keys x_0, x_1
  - x_0 represents '0'
  - x_1 represents '1'
- public key (y_0,y_1):
  - $y_0 = f(x_0)$,
  - $y_1 = f(x_1)$.
  - Where f is a one-way function
- Signature is x_0 if the message is 0 or x_1 if message is 1.
- To sign a message m, use hash and sigh each bit of h(m)



ELLIPTIC CURVE CRYPTOGRAPHY (ECC)

Elliptic curve cryptography (ECC)

- “Elliptic Curve Cryptography” is not a new cryptosystem
- Elliptic curves are a different way to do the math in public key system

- Elliptic curves may be more efficient
- Fewer bits needed for same security

- For equivalent key lengths computations are roughly equivalent
- Hence for similar security ECC offers significant computational advantages

- RFC690: Fundamental Elliptic Curve Cryptography Algorithms

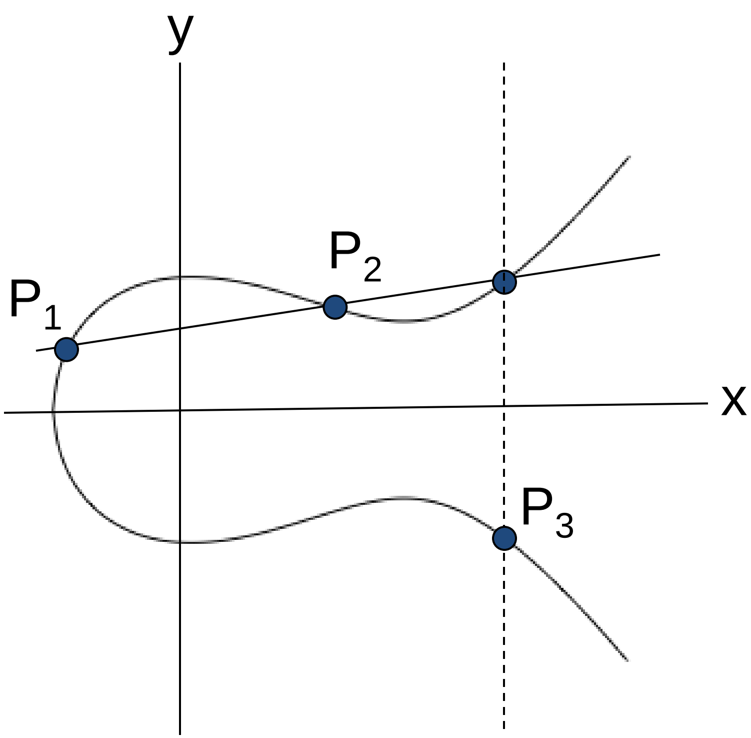
What is an Elliptic Curve?

- An elliptic curve E is the graph of an equation of the form

  $$y^2 = x^3 + ax + b$$

- Also includes a “point at infinity”
- What do elliptic curves look like?
- See the next slide!

Elliptic Curve Picture



- Consider elliptic curve

  E: $y^2 = x^3 - x + 1$

- If P_1 and P_2 are on E, we can define

  $$P_3 = P_1 + P_2$$

  as shown in picture
- Addition is all we need

Points on Elliptic Curve

- Consider $y^2 = x^3 + 2x + 3 \pmod 5$

  $x = 0 \Rightarrow y^2 = 3 \Rightarrow$ no solution (mod 5)

  $x = 1 \Rightarrow y^2 = 6 = 1 \Rightarrow y = 1,4 \pmod 5$

  $x = 2 \Rightarrow y^2 = 15 = 0 \Rightarrow y = 0 \pmod 5$

  $x = 3 \Rightarrow y^2 = 36 = 1 \Rightarrow y = 1,4 \pmod 5$

  $x = 4 \Rightarrow y^2 = 75 = 0 \Rightarrow y = 0 \pmod 5$

- Then points on the elliptic curve are

  (1,1) (1,4) (2,0) (3,1) (3,4) (4,0) and the point at infinity: ∞

Elliptic Curve Math

- Addition on: $y^2 = x^3 + ax + b \pmod{p}$

  $P_1=(x_1,y_1)$, $P_2=(x_2,y_2)$

  $P_1 + P_2 = P_3 = (x_3,y_3)$ where

  $$x_3 = m^2 - x_1 - x_2 \pmod{p}$$

  $$y_3 = m(x_1 - x_3) - y_1 \pmod{p}$$

  And $m = (y_2-y_1)*(x_2-x_1)^{-1} \bmod p$, if $P_1 \neq P_2$

  $m = (3x_1^{\,2}+a)*(2y_1)^{-1} \bmod p$, if $P_1 = P_2$

  Special cases: If m is infinite, $P_3 = \infty$, and

  $\infty + P = P$ for all P

Elliptic Curve Addition

- Consider $y^2 = x^3 + 2x + 3 \pmod 5$. Points on the curve are (1,1) (1,4) (2,0) (3,1) (3,4) (4,0) and ∞
- What is $(1,4) + (3,1) = P_3 = (x_3,y_3)$?

$$m = (1-4)*(3-1)^{-1} = -3*2^{-1}$$

$$= 2(3) = 6 = 1 \pmod 5$$

$$x_3 = 1 - 1 - 3 = 2 \pmod 5$$

$$y_3 = 1(1-2) - 4 = 0 \pmod 5$$

- On this curve, $(1,4) + (3,1) = (2,0)$

Finite Elliptic Curves

- Elliptic curve cryptography uses curves whose variables & coefficients are finite
- have two families commonly used:
  - prime curves $E_p(a,b)$ defined over Z_p
    - use integers modulo a prime
    - best in software
  - binary curves $E_{2m}(a,b)$ defined over $GF(2^n)$
    - use polynomials with binary coefficients
    - best in hardware

Elliptic Curve Cryptography

- ECC addition is analog of modulo multiply
- ECC repeated addition is analog of modulo exponentiation
- need “hard” problem equiv to discrete log
  - `Q=kP`, where Q,P belong to a prime curve
  - is “easy” to compute Q given k,P
  - but “hard” to find k given Q,P
  - known as the elliptic curve logarithm problem
- Certicom example: $E_{23}(9,17)$

ECC Diffie-Hellman

- can do key exchange analogous to D-H
- users select a suitable curve $E_q(a,b)$
- select base point $G=(x_1,y_1)$
  - with large order n s.t. $nG=O$
- A & B select private keys $n_A<n$, $n_B<n$
- compute public keys: $P_A=n_AG$, $P_B=n_BG$
- compute shared key: $K=n_AP_B$, $K=n_BP_A$
  - same since $K=n_An_BG$
- attacker would need to find k, hard

ECC Encryption/Decryption

- several alternatives, will consider simplest
- must first encode any message M as a point on the elliptic curve P_m
- select suitable curve & point G as in D-H
- each user chooses private key $n_A < n$
- and computes public key $P_A = n_A G$
- to encrypt P_m : $C_m = \{ kG, \quad P_m + kP_A \}$, k random
- decrypt C_m compute:

$$P_m + kP_A - n_A(kG) \;=\; P_m + k(n_A G) - n_A(kG) \;=\; P_m$$

ECC Security

- relies on elliptic curve logarithm problem
- fastest method is “Pollard rho method”
- compared to factoring, can use much smaller key sizes than with RSA etc
- for equivalent key lengths computations are roughly equivalent
- hence for similar security ECC offers significant computational advantages

Comparable Key Sizes for Equivalent Security

Symmetric scheme (key size in bits)	**ECC-based scheme (size of *n* in bits)**	**RSA/DSA (modulus size in bits)**
56	112	512
80	160	1024
112	224	2048
128	256	3072
192	384	7680
256	512	15360

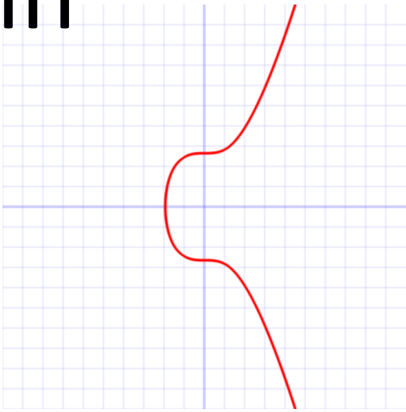
Elliptic curve cryptography (ECC)

- ✓ RFC690: Fundamental Elliptic Curve Cryptography Algorithms
- https://tools.ietf.org/html/rfc6090

- ✓ FIPS PUB 186-4

- Several discrete logarithm-based protocols have been adapted to elliptic curves (replacing the group)

ECC - Example: Bitcoin



- Secp256k1 (with the ECDSA algorithm)
- Parameters (p,a,b,G,n,h)

- The curve E: $y^2 = x^3+ax+b$ over F_p is defined by:
- a = 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
- b = 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000007
- $p = 2^{256} - 2^{32} - 2^9 - 2^8 - 2^7 - 2^6 - 2^4 - 1$

- The base point G in compressed form is:
- G = 02 79BE667E F9DCBBAC 55A06295 CE870B07 029BFCDB 2DCE28D9 59F2815B 16F81798

- and in uncompressed form is:
- G = 04 79BE667E F9DCBBAC 55A06295 CE870B07 029BFCDB 2DCE28D9 59F2815B 16F81798 483ADA77 26A3C465 5DA4FBFC 0E1108A8 FD17B448 A6855419 9C47D08F FB10D4B8

- Finally the order n of G and the cofactor are:
- n = FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE BAAEDCE6 AF48A03B BFD25E8C D0364141
- h = 01

State of the art

Primitive	Parameters	Legacy System Minimum	Future System Minimum
RSA Problem	N, e, d	$\ell(n) \geq 1024$, $e \geq 3$ or 65537, $d \geq N^{1/2}$	$\ell(n) \geq 3072$ $e \geq 65537$, $d \geq N^{1/2}$
Finite Field DLP	p, q, n	$\ell(p^n) \geq 1024$ $\ell(p), \ell(q) > 160$	$\ell(p^n) \geq 3072$ $\ell(p), \ell(q) > 256$
ECDLP	p, q, n	$\ell(q) \geq 160$, $\star$	$\ell(q) > 256$, $\star$
Pairing	p, q, n, d, k	$\ell(p^{k \cdot n}) \geq 1024$ $\ell(p), \ell(q) > 160$	$\ell(p^{k \cdot n}) \geq 3072$ $\ell(p), \ell(q) > 256$

Digital signature

Scheme	Classification: Legacy	Classification: Future
RSA-PSS	✓	✓
ISO-9796-2 RSA-DS2	✓	✓
PV Signatures	✓	✓
(EC)Schnorr	✓	✓
(EC)KDSA	✓	✓
RSA-PKCS# 1 v1.5	✓	✗
RSA-FDH	✓	✗
ISO-9796-2 RSA-DS3	✓	✗
(EC)DSA,(EC)GDSA	✓	✗
(EC)RDSA	✓	✗
ISO-9796-2 RSA-DS1	✗	✗

SOGIS

- The SOG-IS agreement was produced in response to the EU Council Decision of March 31st 1992 (92/242/EEC) in the field of security of information systems, and the subsequent Council recommendation of April 7th (1995/144/EC) on common information technology security evaluation criteria.
- Regarding Cryptography:

https://www.sogis.eu/documents/cc/crypto/SOGIS-Agreed-Cryptographic-Mechanisms-1.3.pdf

SOGIS

- Agreed RSA primitive sizes.

Primitive	Parameters' sizes	R/L	Notes
RSA	$n \geq 3000,\ \log_2(e) > 16$	R	
	$n \geq 1900,\ \log_2(e) > 16$	L [2025]	27-LegacyRSA

SOGIS

- Agreed FF-DLOG Parameters

Family	Group	R/L	Notes
MODP [RFC3526]	3072-bit MODP Group	R	
	4096-bit MODP Group	R	
	6144-bit MODP Group	R	
	8192-bit MODP Group	R	
	2048-bit MODP Group	L[2025]	29-Precomputation, 30-LegacyFF-DLOG
FFDHE [RFC7919]	3072-bit FFDHE Group	R	
	4096-bit FFDHE Group	R	
	6144-bit FFDHE Group	R	
	8192-bit FFDHE Group	R	
	2048-bit FFDHE Group	L[2025]	29-Precomputation, 30-LegacyFF-DLOG

SOGIS

- Agreed Elliptic Curve Parameters

Curve Family	Curve	R/L	Notes
Brainpool [RFC5639]	BrainpoolP256r1	R	
	BrainpoolP384r1	R	
	BrainpoolP512r1	R	
NIST [FIPS186-4, Appendix D.1.2]	NIST P-256	R	34-SpecialP
	NIST P-384	R	
	NIST P-521	R	
FR [JORF]	FRP256v1	R	

SOGIS

- Agreed Asymmetric Encryption Schemes

Primitive	Scheme	R/L	Notes
RSA	OAEP (PKCS#1v2.1) [RFC8017, PKCS1]	R	37-OAEP-PaddingAttack
RSA	PKCS#1v1.5 [RFC8017, PKCS1]	L	36-PaddingAttack

SOGIS

- Agreed Digital Signature Schemes

Primitive	Scheme	R/L	Notes
RSA	PSS (PKCS#1v2.1) [RFC8017, PKCS1, ISO9796-2]	R	
FF-DLOG	KCDSA [ISO14888-3]	R	41-DSARandom
	Schnorr [ISO14888-3]	R	
	DSA [FIPS186-4, ISO14888-3]	R	
EC-DLOG	EC-KCDSA [ISO14888-3]	R	41-DSARandom
	EC-DSA [FIPS186-4, ISO14888-3]	R	
	EC-GDSA [TR-03111]	R	
	EC-Schnorr [ISO14888-3]	R	
RSA	PKCS#1v1.5 [RFC8017, PKCS1, ISO9796-2]	L	40-PKCSFormatCheck


Questions?