Digital signatures

GIANLUCA DINI
Dept. of Ingegneria dell'Informazione
University of Pisa
email: gianluca.dini@unipi.it
Version: 2024-04-18

1

Digital Signatures

OVERVIEW

Apr-24 Digital signatures 2

2

The problem →

- Alice and Bob share a secret key k
- Alice receives and decrypts a message which makes semantic sense
- Then, Alice concludes that the message comes from Bob
- Message origin authetication ➔ message integrity
  - Beware, we know that ciphers are malleable!
- MDC / MAC does not change the reasoning

The problem →

- The reasoning above works under the assumption of mutual trust
  - If a dispute arise, Alice cannot prove to a third party that Bob generated the message
- There are practical cases in which Alice and Bob wish to securely communicate but they don’t trust each other
  - E.g., e-commerce: customer and merchant have conflicting interests

The problem

↓

- Provability/verifiability requirement
  - If a dispute arises an unbiased third party must be able to solve the dispute equitably, without requiring access to the signer's secret
- Symmetric cryptography is of little help
  - Alice and Bob have the same knowledge and capabilities
- Public-key cryptography is the solution
  - Make it possible to distinguish the actions performed by who knows the private key

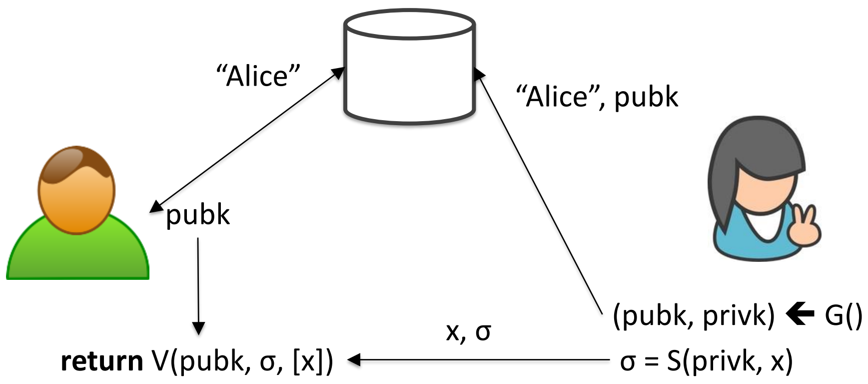
5

Digital signature scheme

- A signature scheme is defined by three algorithms
- Key generation algorithm G
  - takes as input 1^n and outputs (pubk, privk)
- Signature generation algorithm S
  - takes as input a private key privk and a message x and outputs a signature σ = S(privk, x)
- Signature verification algorithm V
  - takes as input a public key pubk, a signature σ and (optionally) a message x and outputs True or False

6

Communication model



[*optional field*]

Apr-24 Digital signatures 7

7

Properties

- Consistency Property
  - For all x and (pubk, privk), V(pubk, [x] S(privk, x)) = TRUE
- Security property (informal)
  - Even after observing signatures on multiple messages, an attacker should be unable to forge a valid signature on a new message

Apr-24 Digital signatures 8

8

Threat model

- Adaptive chosen-message attack
  - The attacker is able to induce the sender to sign *messages of the attacker's choice*
  - The attacker knows the public key
- Existential unforgeability (security goal/req)
    - Attacker should be *unable* to forge valid signature on *any* message not signed by the sender

9

Security property implies...

- Integrity
- Verifiability
- Non-repudiation
- No confidentiality
  - Use a cipher (AES, 3DES,…) if confidentiality is a requirement

10

Algorithm families

- Integer factorization
  - RSA
- Discrete logarithm
  - ElGamal, DSA
- Elliptic curves
  - ECDSA

11

Digital signatures

NON-REPUDIATION VS AUTHENTICATION

12

Non-repudiation

- Non-repudiation prevents a signer from signing a document and subsequently being able to successfully deny having done so.

Non-repudiation vs authentication

- Authentication
  - Based on symmetric cryptography
  - Allows a party to convince itself or a mutually trusted party of the integrity/authenticity of a given message at a given time t_0
- Non-repudiation
  - based on public-key cryptography
  - allows a party to convince others at any time $t_1 \geq t_0$ of the integrity/authenticity of a given message at time t_0

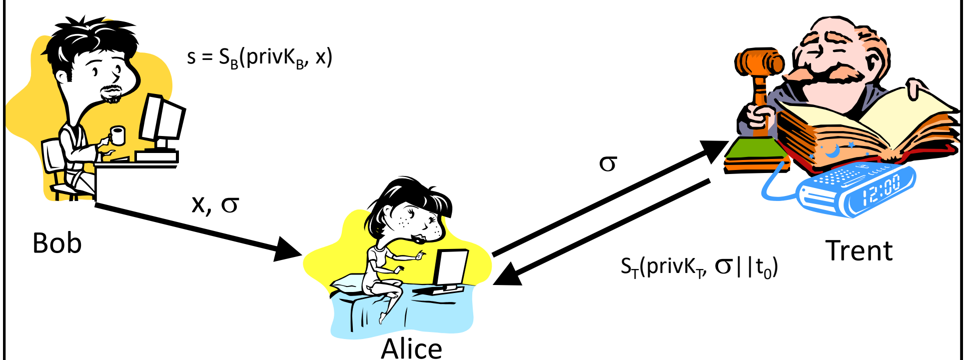
Dig sig vs non-repudiation

- Data origin authentication as provided by a digital signature is valid only while the secrecy of the signer's private key is maintained
- A threat that must be addressed is a signer who intentionally discloses his private key, and thereafter claims that a previously valid signature was forged
- This threat may be addressed by
  - Prevent direct access to the key
  - Use of a trusted timestamp agent
  - Use of a trusted notary agent

Apr-24 Digital signatures 15

15

Trusted Timestamping Service



- Trent certifies that digital signature σ *exists* at time t_0
- If Bob's $privk_B$ is compromised at $t_1 > t_0$, then σ is valid

Apr-24 Digital signatures 16

16

Trusted Notary Service

- Trent certifies that a *certain statement* on the digital signature s is true at a certain time t0
  - TNS generalize the TTS
  - Examples of statements
    - Signature s exists at time t0
    - Signature s is valid at time t0
- Trent may certify the existence of a certain document
  - t = H(document), σ = S(privKT, t || timestamp)
  - Document remains secret
- Trent is trusted to verify the statement before issuing it

Apr-24 Digital signatures 17

17

Digital Signatures

COMPARISON TO MAC

Apr-24 Digital signatures 18

18

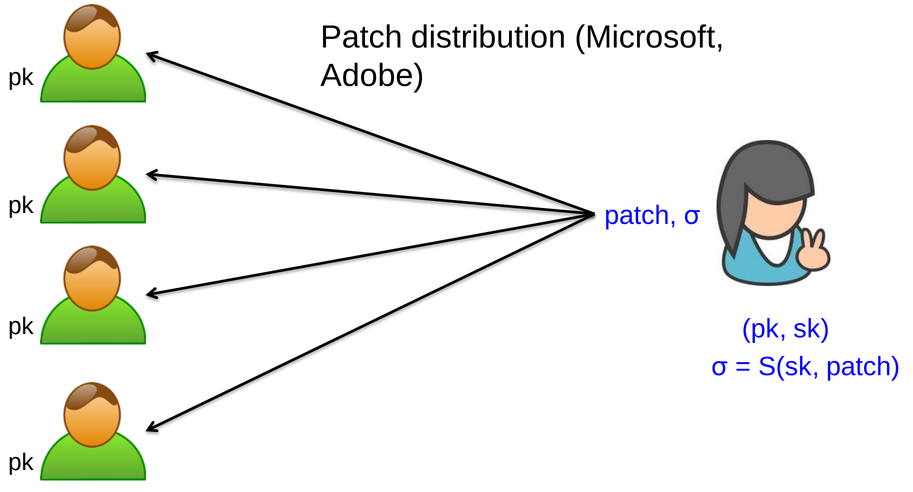
Digital signatures

- Provide *integrity* in the public-key setting
- Analogous to message authentication codes (MACs) but some key differences…

19

Prototypical application

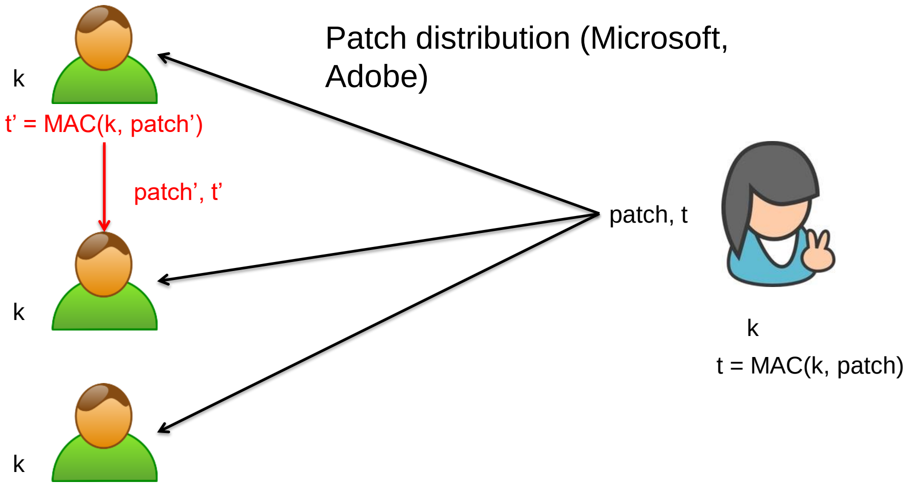
Patch distribution (Microsoft, Adobe)



20

Comparison to MACs

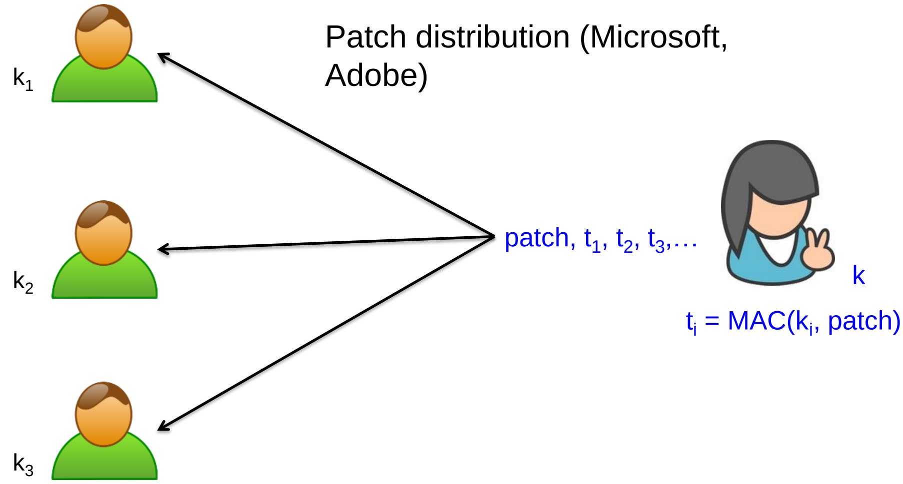
Patch distribution (Microsoft, Adobe)



21

Comparison to MACs

Patch distribution (Microsoft, Adobe)



22

Comparison to MACs

- Single shared key k
  - A client may forge the tag
  - Unfeasible if clients are not trusted
- Point-to-point keys k_i
  - Computing and network overhead
  - Prohibitive key management overhead
  - Unmanageable!

Apr-24 Digital signatures 23

23

Comparison to MACs

- Public verifiability
  - Dig Sig: anyone can verify the signature
  - MAC: Only a holder of the key can verify a MAC tag
- Transferability
  - Dig Sig can forward a signature to someone else
  - MAC cannot

%

Apr-24 Digital signatures 24

24

Comparison to MACs

- Non-repudiability
  - Signer cannot (easily) deny issuing a signature
    - Crucial for legal application
    - Judge can verify signature using a copy of pK
  - MACs cannot provide this functionality
    - Without access to the key, no way to verify a tag
    - Even if receiver leaks key to judge, how can the judge verify the key is correct?
    - Even if the key is correct, receiver could have generated the tag!

Apr-24 Digital signatures 25

25

Digital signatures

THE RSA SIGNATURE SCHEME

Apr-24 Digital signatures 26

26

Plain RSA

- Key generation
  - (e, n) public key; (d, n) private key
- Signing operation
  - $\sigma = x^d \bmod n$
- Verification operation
  - Return ($x == \sigma^e \bmod n$)

Properties

- Computational aspects
  - *The same considerations as PKE*
- Security
  - Algorithmic attacks
    - Factoring
  - Existential forgery
  - Malleability

Existential forgery

- Given public key (n, e), generate a valid signature for a random message x
  - Choose a signature σ
  - Compute $x = \sigma^e \bmod n$
  - Output (x, σ)
    - It turns out that σ is positively verified as the digital signature of x
  - Message x is random and may have no application meaning.
  - However, this property is highly undesirable

Apr-24 Digital signatures 29

29

Malleability

- Combine two signatures to obtain a third (existential forgery)
  - Exploit the homomorphic property of RSA
- The attack
  - Given $\sigma_1 = x_1^d \bmod n$
  - Given $\sigma_2 = x_2^d \bmod n$
  - Output $\sigma_3 \equiv (\sigma_1 \bullet \sigma_2) \bmod n$ that is a valid signature of $x_3 \equiv (x_1 \bullet x_2) \bmod n$
    - PROOF. $x_3 = \sigma_3^e \equiv (\sigma_1 \bullet \sigma_2)^e \equiv \sigma_1^e \bullet \sigma_2^e \equiv x_1^{de} \bullet x_2^{ed} \equiv x_1 \bullet x_2 \bmod n$

Apr-24 Digital signatures 30

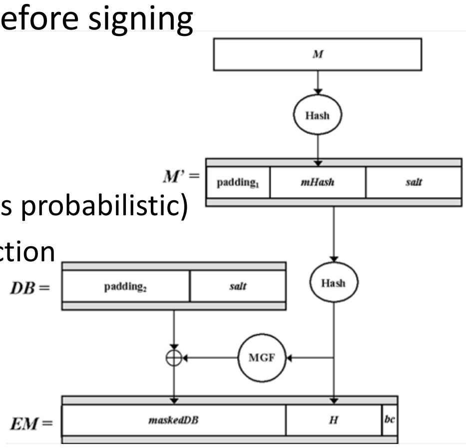
30

RSA Padding

- Plain RSA is never used
  - Because of existential forgery and malleability,
- Padding
  - Padding allows only certain message formats
    - It must be difficult to choose a signature whose corresponding message has that format
  - Probabilistic Signature Scheme in PKCS#1
    - Encoding Method for Signature with Appendix (EMSA)

PSS

- The message is encoded before signing
  - s = EM^d mod n where
  - M = message
  - EM = encoded message
  - salt : random value (makes s probabilistic)
  - MGF: mask generation function
  - fixed values:
    - bc, padding1, padding2



Digital Signatures

DIGITAL SIGNATURES VS HASH FUNCTIONS

Signing long messages

- Consider RSA digsig
  - Message $0 \leq x < n$
    - E.g., n = 1024–3072 bits (128–384 bytes)
  - What if $x > n$?
  - An ECB-like approach is not recommended
    1. High-computational load (performance)
    2. Message overhead (performance)
    3. Block reordering and substitution (security)
- We would like to have a short signature for messages on any length
- The solution of this problem is hash functions

Dig sig vs hash properties

- Hash functions properties
  - Pre-image resistance
  - Second pre-image resistance
  - Collision resistance
- These properties are crucial for digital signatures security

Apr-24 Digital signatures 35

35

Dig sig vs hash properties

- Pre-image Resistance
  - Digital signature scheme based on (school-book) RSA
    - (n, d) is Alice's private key;
    - (n, e) is Alice's public key
    - Tag $t = H(x)$, $s = t^d \pmod n$
  - If H is not pre-image resistant, then existential forgery is possible
    - Select $z < n$
    - Compute $y = z^e \bmod n$
    - Find x' such that $H(x') = y$ (🡐)
    - Claim that z is the digital signature of m' Q.E.D

Apr-24 Digital signatures 36

36

Dig sig vs hash properties

- 2nd preimage resistance
  - The protocol
    - Bob → Alice: x
    - Alice → Bob: x, s = S(privK_A, t) with t = H(x)
  - If H is not 2nd-preimage resistant, the following attack is possible
    - An adversary (e.g., Alice herself) can determine a 2nd-preimage x' of x and then (🡐)
    - Then claim that Alice has signed x’ instead of x Q.E.D

Apr-24 Digital signatures 37

37

Dig sig vs hash properties

- Collision-resistance
  - If H is not collision resistant, the following attack is possible
    - Alice chooses x and x’ s.t. H(x) = H(x’) (🡐)
    - computes s = S(privK_A, H(x))
    - Sends (x, s) to Bob
    - later claims that she actually sent (x', s) Q.E.D

Apr-24 Digital signatures 38

38

Hash-and-Sign paradigm

- Given a signature scheme Σ = (G, S, V) for “short” messages of length n-bit
- Given a Hash function H: $\{0, 1\}^* \rightarrow \{0, 1\}^n$
- Construct a signature scheme Σ' = (G, S’, V’) for messages of any length
  - σ = S’(privK, m) = S(privk, H(m))
  - V’(m, pubK, σ) = V(H(m), pubK, σ)

Apr-24 Digital signatures 39

39

Hash-and-sign paradigm

- THM. If Σ is secure and H is collision-resistant, then Σ' is secure
  - PROOF by contradiction
    1) Assume that the sender authenticates m_1, m_2,…
    2) Assume the sender manages to forge (m’, σ’), m’ ≠ m_i, for all i
    3) Let h_i = H(mi). Then, we have two cases
       1) If H(m’) = h_i for some i, then collision in H (contradiction)
       2) If H(m') ≠ h_i, for all i, then forgery in Σ (contradiction)

Apr-24 Digital signatures 40

40

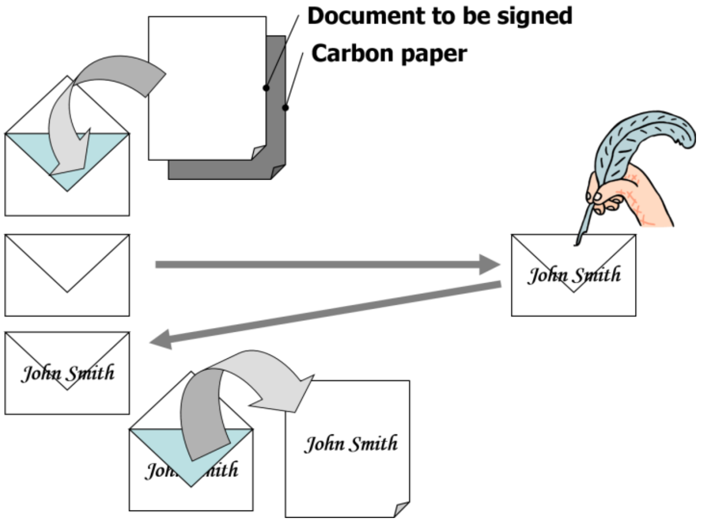
Digital signatures

RSA-BASED BLIND SIGNATURES

Blind signatures

- Intuition
  - In a blind signature scheme, the signer can't see what it is signing
- Unlinkabiliy
  - The signer is not able to link the signature to the act of signing

The metaphor



43

Blind signatures

→

- The protocol
  1. Alice
     a) Randomly chooses b s.t. gcd(b, n) = 1
     b) Computes $x' \equiv x \cdot b^e \pmod{n}$
     c) Sends x' to Bob (signer)
  2. Bob
     a) Receive x'
     b) Compute $s' \equiv (x')^d \pmod{n}$
     c) Returns s' Alice

44

Blind signatures →

- The protocol
  3. Alice
     a) Receive s’
     b) Compute s, the digital signature of x, $s \equiv s' \cdot b^{-1} \pmod n$
- Proof
  – $s' \cdot b^{-1} \equiv (x')^d \cdot b^{-1} \equiv (x \cdot b^e)^d \cdot b^{-1} \equiv x^d \cdot b^{ed} \cdot b^{-1} \equiv$
  $\equiv x^d \cdot b \cdot b^{-1} \equiv x^d \equiv s \bmod n$ QED

Apr-24 Digital signatures 45

45

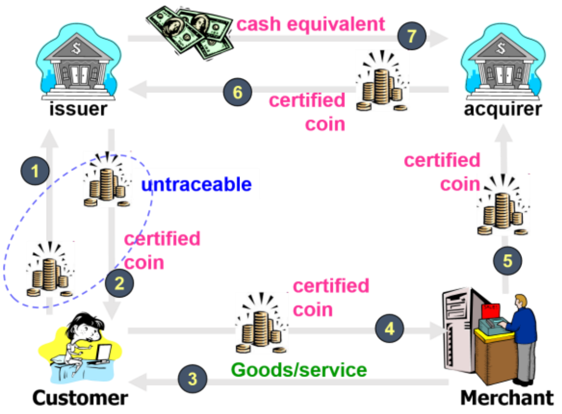
Applications

- Privacy related applications
  – Digital cash
    - Chaum, David (1983). "Blind Signatures for Untraceable Payments." Advances in Cryptology.
  – Electronic voting

Apr-24 Digital signatures 46

46

Digital cash

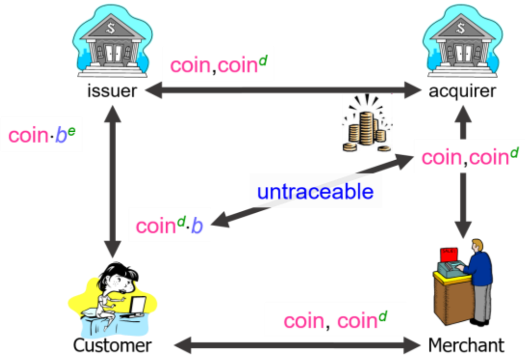


- coin: a random number
- coin·b^e: blinded coin
- coin, $coin^d$: certified coin
- $d_{10€}$: a 10€ worth bank's private key

Apr-24 Digital signatures 47

47

Digital cash



- coin: a random number
- coin·b^e: blinded coin
- coin, $coin^d$: certified coin
- $d_{10€}$: a 10€ worth bank's private key

Apr-24 Digital signatures 48

48

Double spending →

- The protocol does not prevent double spending
  - the customer can spend the digital coin multiple times
  - The merchant can deposit the digital coin multiple times
- Partial countermeasure
  - The issuer maintains the list of spent digital coins
    - Protect the bank from frauds
    - Don’t allow issuer to identify the fraudster

Apr-24 Digital signatures 49

49

Double spending ↓

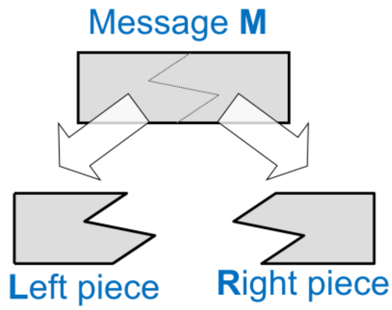
- Purely criptographic solutions based on
  - Secret splitting
  - Bit commitment
  - Cut-and-choose
  - Inefficient but great impulse to cryptography
- Hardware solutions
  - The Mondex smart card e-cash system
    - 90’s technology; never left the experimental phase
- Bitcoin and blockchain

Apr-24 Digital signatures 50

50

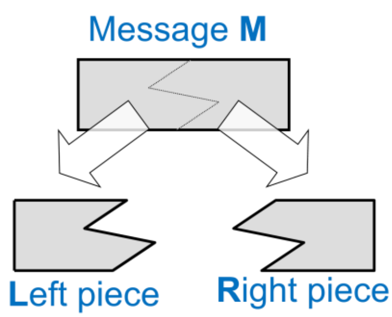
Secret splitting [→]

- Each piece alone gives no information on the message
- Both pieces make it possible to reconstruct the message



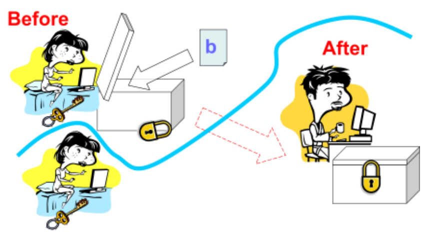
Secret splitting

- EXAMPLE
- Creating L and R
  - Message **M**
  - **R** ← random()
  - **L** = **M** ⊕ **R**
- Message reconstruction
  - M = L ⊕ R



Bit commitment [→]

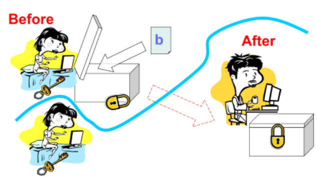
- Alice thinks of a number and Bob has to guess it.
- Alice thinks about the number but doesn't want to reveal it.
- Bob guesses the number but wants to be sure Alice doesn't change it.



Apr-24 Digital signatures 53

53

Bit Commitment [→]



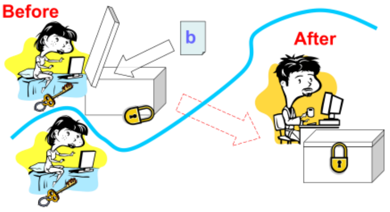
- Perfectly binding
  - It is theoretically impossible for Alice to alter her commitment after she makes it
- Perfectly concealing
  - It is theoretically impossible for Bob to find commitment without Alice revealing it
- THM There exists no commitment scheme which is both perfectly binding and perfectly hiding

Apr-24 Digital signatures 54

54

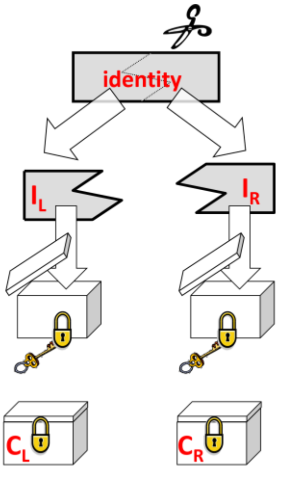
Bit Commitment: toy example

- Example (Perfectly binding)
  - Parameters
    - p: large prime
    - g: a generator
  - Commitment phase
    - Alice randomly selects b in [0, p – 1]
    - Alice computes commitment $c = g^b \bmod p$
    - Alice publishes c
  - Reveal Phase
    - Alice publishes p
    - Bob checks whether $c == g^b \bmod p$
  - Not perfectly concealing as $\leq_P$ DLP.



55

On solving double spending



56

On solving double spending

- Coin = [coin, identity string, $h(\text{coin, identity string})^d$]
- Uniqueness bit string: coin ← random()
- Identity bit strings
  - $I_i \to \langle I_{iL}, I_{iR} \rangle$
  - $(C1_L, C_{1R}), (C_{2L}, C_{2R}), \ldots, (C_{100L}, C_{100R})$
  - Pairs are different from each other
- Setup (money order)
  - Alice prepares 100 blank coin

On solving double spending: cut-and-choose

$H(\text{coin}_i, \text{identity string}_i) \times b_i^e$, $1 \le i \le 100$ →

← unblind 99 chosen at random)

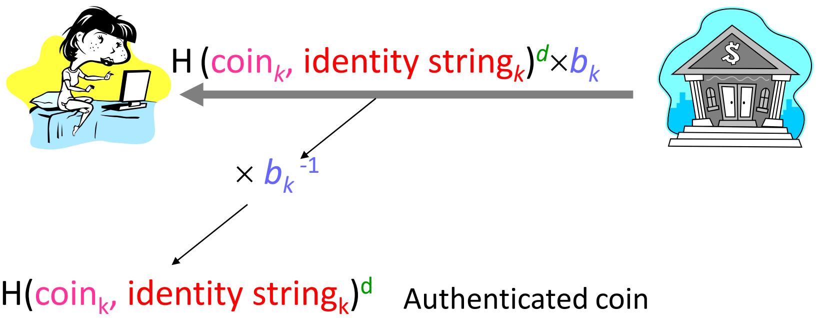
$H(\text{coin}_i, \text{identity string}_i)$, $1 \le i \le 99$ →

← Open 99 identity strings

$((I_{jL}, I_{jR})_i, 1 \le j \le 100), 1 \le i \le 99$ →

At the end of the protocol, the bank is 99% convinced that the undisclosed commitment contains Alice's identity

On solving double spending: withdraw

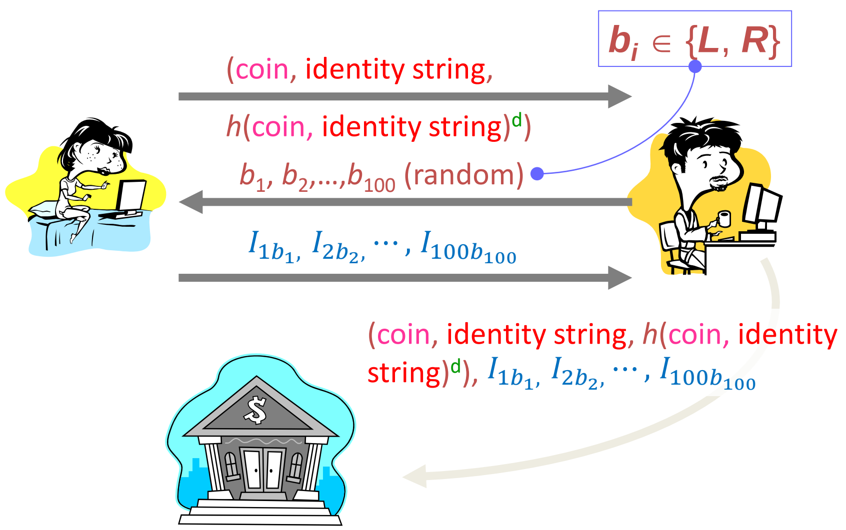


The bank "signs" the "blank" coin that is left over (e.g., the k-th)

Apr-24 Digital signatures 59

59

On solving double spending: spend



Apr-24 Digital signatures 60

60

On solving double spending: bank's controls

1. The bank verifies the digital signature
2. If the coin has not yet been spent
    1. the bank credits an amount equal to the denomination to Bob
3. Otherwise (double spending)
    1. if the identity strings are the same
        1. then the fraudster is the merchant Bob;
    2. otherwise
        1. the fraudster is Alice

On solving double spending: fraudster detection

- In case the coin has already been spent
- If the identity strings are the same, then the fraudster is Bob, otherwise
- If the identity strings are different, then the fraudster is Alice
    - The bank finds a position in the identity string where Alice has revealed the right and left pieces of her identity with probability $1 - (½)^{100}$
    - From the two pieces the bank determines Alice's identity

Digital signatures

THE ELGAMAL SIGNATURE SCHEME

Apr-24 Digital signatures 64

64

Elgamal in a nutshell

- Invented in 1985
- Based on difficulty of discrete logarithm
- Digital signature operations are different from the cipher operations

Apr-24 Digital signatures 65

65

Key generation

- Choose a large prime *p*
- Choose a primitive element α of (a subgroup of) $\mathbb{Z}_p^*$
- Choose a random number $d \in \{2, 3, \ldots, p - 2\}$
- Compute $\beta = \alpha^d \bmod p$
- pubK = (p, α, β)
- privK = d

Apr-24 Digital signatures 66

66

Signature generation

- Input message x
- Choose an ephemeral key k_E in {0, 1, 2, p – 2} such that $\gcd(k_E, p - 1) = 1$
- Compute the signature parameters
  - $r \equiv \alpha^{kE} \bmod p$
  - $s \equiv (x - d \cdot r)k_E^{-1} \bmod p - 1$
  - (r, s) is the digital signature
- Output $\langle x, (r, s)\rangle$

Apr-24 Digital signatures 67

67

Signature verification

- Let
  - (p, α, β) be the public key;
  - x be the message and
  - (r, s) be the digital signatire
- Compute $t \equiv \beta^r \cdot r^s \bmod p$
- If ($t \equiv \alpha^x \bmod p$) ➔ valid signature; otherwise ➔ invalid signature

Apr-24 Digital signatures 68

68

Proof

1. Let $t \equiv \beta^r \cdot r^s \equiv (\alpha^d)^r (\alpha^{kE})^s \equiv \alpha^{d \cdot r + kE \cdot s} \bmod p$
2. If $\beta^r \cdot r^s \equiv \alpha^x \bmod p$ then $\alpha^x \equiv \alpha^{d \cdot r + kE \cdot s} \bmod p$ [Eq. a]
3. According to Fermat's Little Theorem Eq.a holds if $x \equiv d \cdot r + k_E \cdot s \bmod p - 1$
4. from which the construction of parameter $s = (x - d \cdot r) k_E^{-1} \bmod p - 1$

Apr-24 Digital signatures 69

69

Computational aspects

- Key generation
  - Generation of a large prime (1024 bits)
  - True random generator for the private key
  - Exponentiation by square-and-multiply
- Signature generation
  - $|s| = |r| = |p|$ thus $|x, (r, s)| = 3\ |x|$ (*dig sig expansion*)
  - One exponentiation by square-and-multiply
  - One inverse k_E^{-1} mod p by EEA (pre-computation)
- Signature verification
  - Two exponentiations by square-and-multiply
  - One multiplication

Apr-24 Digital signatures 70

70

Security aspects

- The verifier must have the correct public key
- The DLP must be intractable
- *Ephemeral key K_E cannot be reused* (➔)
  - If K_E is reused the adversary can compute the private key *d* and impersonate the signer
- Existential forgery for a random message x unless it is hashed (➔)

Apr-24 Digital signatures 71

71

Reuse of ephemeral key

- If the ephemeral key k_E is reused, an attacker can easily compute the private key d
  - Proof
    - Message x_1 and x_2 and the reused ephemeral key k_E
    - $(x_1, (s_1, r))$ and $(x_2, (s_2, r))$ where $r \equiv \alpha^{kE} \bmod p$
      - $s_1 \equiv (x_1 - d \cdot r) \cdot k_E^{-1} \bmod p - 1$ [Eqn. a]
      - $s_2 \equiv (x_2 - d \cdot r) \cdot k_E^{-1} \bmod p - 1$ [Eqn. b]
      - *Eqn.a and Eqn.b is a system in two unknowns* (k_E and d) and two equations
      - $s_1 - s_2 \equiv (x_1 - x_2) \cdot k_E^{-1} \bmod p - 1$
      - $k_E \equiv (x_1 - x_2) \cdot (s_1 - s_2)^{-1} \bmod p - 1$
      - $d \equiv (x_1 - s_1 \cdot k_E) \cdot r^{-1} \bmod p - 1$ Q.E.D.

Apr-24 Digital signatures 72

72

Existential Forgery Attack [→]

- The attack

Alice	Adversary	Bob
		privK = d, pubK = (p, α, β)
	< ---------------(p, α, β)-------------------	
	1. select i, j, s.t. gcd(j, p – 1) = 1	
	2. compute the signature	
	$r \equiv \alpha^i \cdot \beta^j \bmod p$	
	$s \equiv -r \cdot j^{-1} \bmod p - 1$	
	3. compute the message	
	$x \equiv s \cdot i \bmod p - 1$	
verification	< -----------------(x, (r, s))---------------	
$t \equiv \beta^r \cdot r^s \bmod p$ since		
$t \equiv \alpha^x \bmod p$ → valid signature!		

Apr-24 Digital signatures 73

73

Existential forgery attack

- Proof

$t \equiv \beta^r \cdot r^s \equiv (\alpha^d)^r \cdot (\alpha^i \cdot \beta^j)^s \equiv (\alpha^d)^r \cdot (\alpha^i \cdot \alpha^{d \cdot j})^s \equiv \alpha^{d \cdot r} \cdot (\alpha^{i+d \cdot j})^s$

$\equiv \alpha^{d \cdot r} \cdot (\alpha^{i+d \cdot j})^s \equiv \alpha^{d \cdot r} \cdot \alpha^{(i+d \cdot j)\cdot(-r \cdot j^{-1})} \equiv$

$\equiv \alpha^{d \cdot r} \cdot \alpha^{-d \cdot r} \cdot \alpha^{-r \cdot i \cdot j^{-1}} \equiv \alpha^{s \cdot i} \bmod p$ **[Eqn. a]**

- As the message was constructed as $x \equiv s \cdot i \bmod p$ then Equation a $\alpha^{s \cdot i} \equiv \alpha^x \bmod p$ which is the condition to accept the signature as valid
- In Step 3, the adversay computes message x whose semantics (s)he cannot control
- The attack is not feasible if the message is hashed
  - $s \equiv (H(x) - d \cdot r) k_E^{-1} \bmod p - 1$

Apr-24 Digital signatures 74

74

Digital Signatures

DIGITAL SIGNATURE ALGORITHM (DSA)

Apr-24 Digital signatures 75

75

Introduction

- The Elgamal scheme is rarely used in practice
- DSA is a more popular variant
  - It’s a federal US government standard for digital signatures (DSS)
  - It was proposed by NIST
- Advantages of DSA w.r.t. Elgamal
  - Signature is only 320 bits
  - Some attacks against Elgamal are not applicable to DSA

Key Generation

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 ord(α) = q, i.e., α generates the *subgroup with q elements*.
4. Choose a random integer d with 0 < d < q.
5. Compute $\beta \equiv \alpha^d \bmod p$.
6. The keys are now:
   1. pubK = (p,q,α,β)
   2. privK = (d)

Central idea

- DSA uses two cyclic groups
  - $\mathbb{Z}_p^*$, the order of which has bit lenght 2014 bit
  - H_q, a 160-bit subgroup of $\mathbb{Z}_p^*$
  - This setup yields shorter signatures
- Other combinations are possible

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

Signature Generation

1. Choose an integer as random ephemeral key k_E with $0 < k_E < q$.
2. Compute $r \equiv (\alpha^{kE} \bmod p) \bmod q$.
3. Compute $s \equiv (SHA(x) + d \cdot r)k_E^{-1} \bmod q$.
   - SHA-1(·) produces a 160-bit value
4. Digital signature is the pair (r, s)
   - 160 + 160 = 320 bit long

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 $v \equiv (\alpha^{u1} \cdot \beta^{u2} \bmod p) \bmod q$.
5. The verification follows from:
   1. If $v \equiv r \bmod q$ ➔ valid signature
   2. Otherwise ➔ invalid signature

80

Proof [→]

- We show that a signature (r, s) satisfies the verification condition $v \equiv r \bmod q$.
  - $s \equiv (SHA(x)+d\,r)k_E^{-1} \bmod q$ which is equivalent to $k_E \equiv s^{-1}\,SHA(x)+d\,s^{-1}\,r \bmod q$.
  - The right-hand side can be expressed in terms of the auxiliary values u1 and u2: $k_E \equiv u_1+d\,u_2 \bmod q$.
  - We can raise α to either side of the equation if we reduce modulo p: $\alpha^{kE} \bmod p \equiv \alpha^{u1 + d\,u2} \bmod p$

[→]

81

Proof

- Since the public key value β was computed as $\beta \equiv \alpha^d \bmod p$, we can write: $\alpha^{kE} \equiv \alpha^{u1} \beta^{u2} \bmod p$.
- We now reduce both sides of the equation modulo q: $(\alpha^{kE} \bmod p) \bmod q \equiv (\alpha^{u1}\beta^{u2} \bmod p) \bmod q$.
- Since r was constructed as $r \equiv (\alpha^{kE} \bmod p) \bmod q$ and $v \equiv (\alpha^{u1}\beta^{u2} \bmod p) \bmod q$,
- this expression is identical to the condition for verifying a signature as valid: $r \equiv v \bmod q$.

Apr-24 Digital signatures 82

82

Computational aspects [→]

- Key Generation
  - The most challenging phase
    - Find a $\mathbb{Z}_p^*$ with 1024-bit prime *p* and a subgroup in the range of 2^{160}
    - This condition is fulfilled if $|\ \mathbb{Z}_p^*\ | = |p - 1|$ has a prime factor q of 160 bit
  - General approch:
    - To find q first and then p

Apr-24 Digital signatures 83

83

Computational aspects [→]

- Signing
  - Computing r requires exponentiation
    - Operands are on 1024 bit
    - Exponent q is on 160 bit
      - On average 160 + 80 = 240 SQs and MULTs
    - Result is reduced mod q
    - Does not depend on message x so can be precomputed
  - Computing s
    - Involve 160-bit operands
    - The most costly operation is inverse

Apr-24 Digital signatures 84

84

Computational aspects

- Verification
  - Computing the auxiliary parameters w, u_1 and u_2 involves 160-bit operands
  - This is relatively fast

Apr-24 Digital signatures 85

85

Security

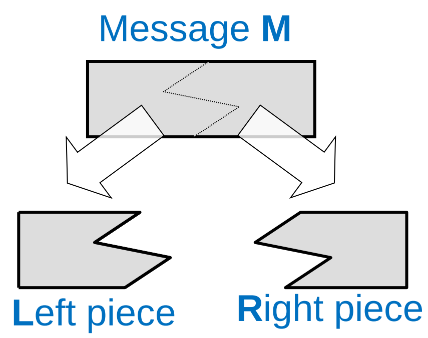
- We have to protect from two different DLPs
  1. d = $\log_{\alpha} \beta$ mod p.
     - Index calcolus attack
       - Prime p must be on 1024 bits for 80-bit security level
  2. α generates a subgroup of order q
     - Index calculus attack cannot be applied
     - Only generic DLP attacks can be used
       - Square-root attacks: Baby-step giant-step, Pollard's rho
       - Running time: $\sqrt{q} = \sqrt{2^{160}} = 80$
- Vulerable to k_E reuse
  - Analalogue to ElGamal

Apr-24 Digital signatures 86

86

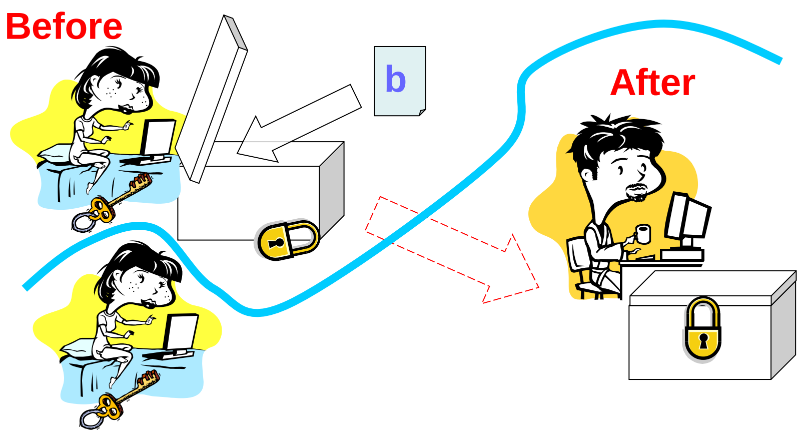
Apr-24 Digital signatures 87

87



Apr-24 Digital signatures 88

88



Apr-24 Digital signatures 89

89