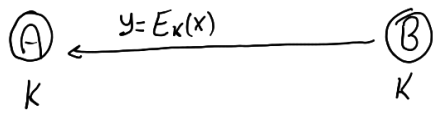
Digital signatures

GIANLUCA DINI
Dept. of Ingegneria dell'Informazione
University of Pisa
email: gianluca.dini@unipi.it
Version: 01/04/2025

1



A, B share a secret key.
Suppose Bob sends a mess. to Alice encrypted by means of symm. encryption.
Assume message makes semantical sense (she got what she was expecting)

Digital Signatures

OVERVIEW

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! (ignore it for now)
- MDC / MAC does not change the reasoning

↑ hash or MAC do not change reasoning. If receiver can verify that message is the expected one, they can say message is coming from communicating peer because they know shared secret.

3

The problem →

↳ I assume other guy does not reveal secret and that he is skilled (he can manage secrets)

- 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 : if Bob says "I never said Y", Alice cannot disprove it*
- 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

* Same message can be generated by her too.

4

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

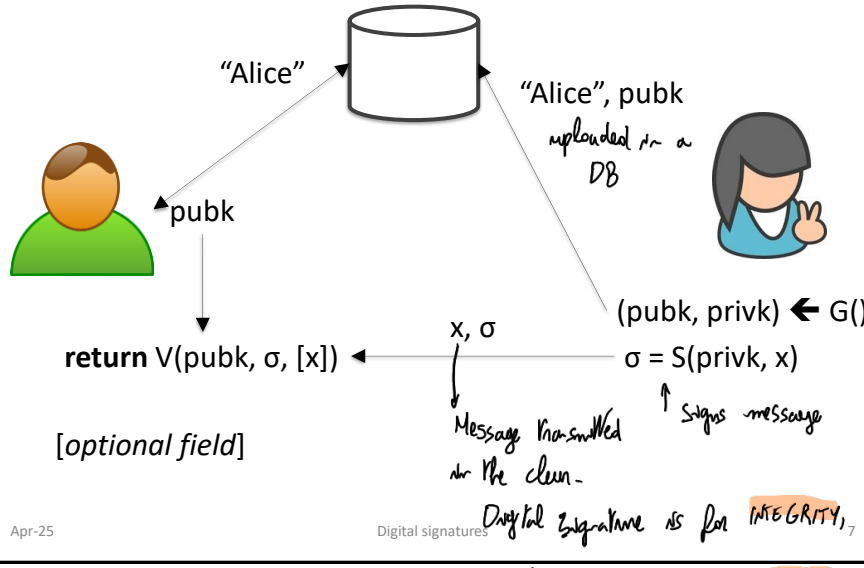
5

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



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 (computationally impossible)

Threat model

- Adaptive chosen-message attack
  - The attacker is able to induce the sender to sign *messages of the attacker's choice* (Not only able to collect pairs, but make them sign)
  - 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

Apr-25 Digital signatures 9

9

Security property implies...

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

Apr-25 Digital signatures 10

10

Algorithm families

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

ElGamal DS is a variation of El Gamal Cypher.
DSA is a variation of the El Gamal to make it more efficient

↳ EC redefinition of DSA.

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.

13

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 example we started from. At the moment Alice receives a message she can prove authenticity*
- Non-repudiation
  - based on public-key cryptography
  - allows a party to convince *others* (not only comm. part.) at any time $t_1 \geq t_0$ of the integrity/authenticity of a given message at time t_0

*Why not any time? Because symmetric cryptography you don't store the message to provide evidence later on. You decrypt it and that's it.

14

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 (1)

↳ scenario in which mutual trust is not present

15

Dig sig vs non-repudiation [→]

- The threat may be addressed by preventing direct access to the key
  - Use of a Hardware Security Module (HSM) Generated on board, with no operation to retrieve it from HW.
  - (2) Use of a Trusted Third-Party Key Management
  - Use of Trusted timestamp agent or audit trail
  - Use of threshold cryptography *

There are applications in which you have access to your private key

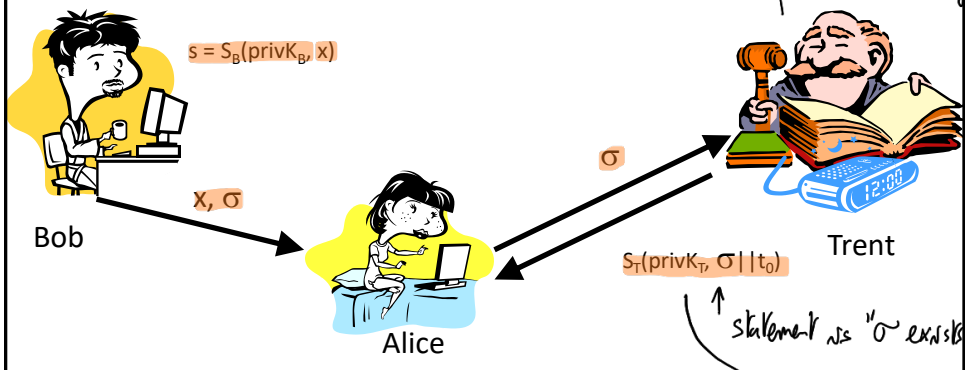
* Split key to N pieces, each to different server and I need that each server contributes to the signature. If 5/10 servers give OK then I have signature. → Those are parameters

16

- If my key is stored on a file in the HDD. A virus could send the file somewhere and perform a password attack. You only get to know of what happened because of the effects and consequences. While with a smartcard, you cannot retrieve key. Only kind of attacks are checking you to sign something and use that.
For a smartcard, untrusted party should say that they have lost smartcard without revealing. You must be very convincing! ☺

② University of Pisa, for example. A server manages private key on your behalf.

Trusted Timestamping Agent



- 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

17

Trusted Notary Agent

- TNS generalizes the TTS
  - Trent certifies that a *certain statement* on the digital signature σ is true at a certain time t_0
  - Trent is trusted to verify the statement before issuing it
  - Examples of statements
    - Signature σ **exists** at time t_0
    - Signature σ is **valid** at time t_0
- Example:
  - Trent may certify the existence of a certain document
  - tag = H(document), σ = S($privK_T$, tag || timestamp)
  - The document remains secret

18

Digital Signatures

COMPARISON TO MAC

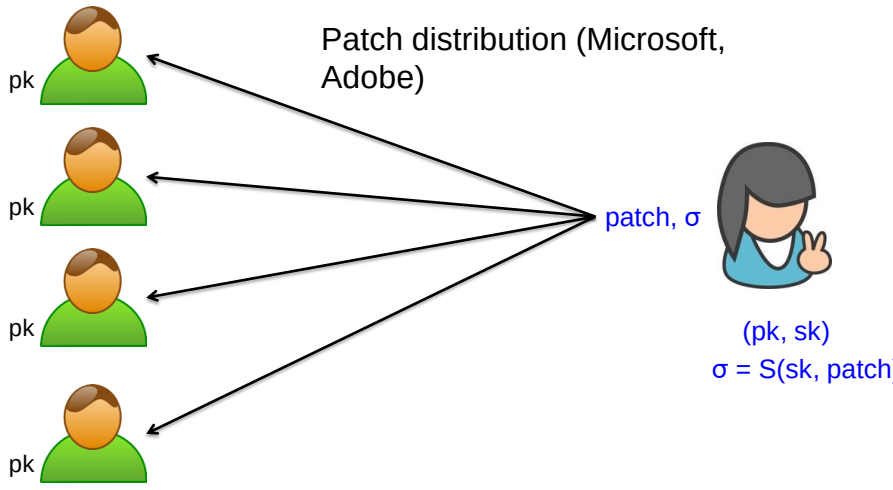
19

Digital signatures

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

20

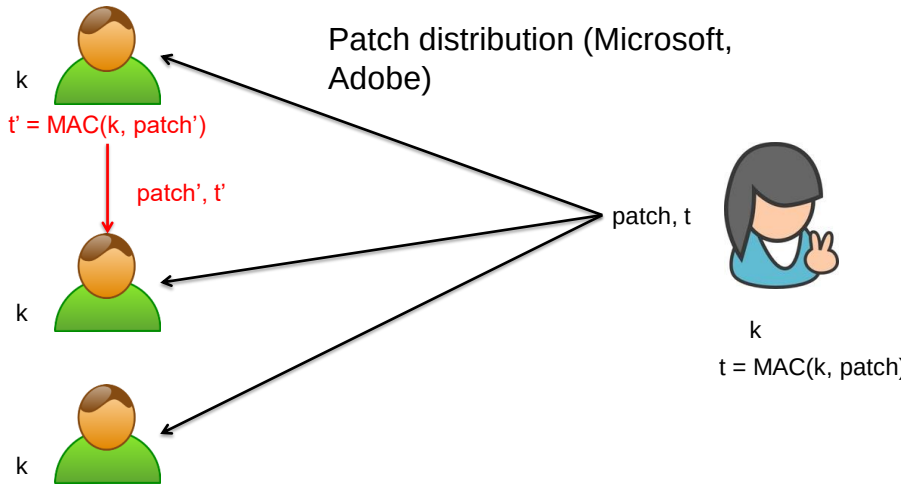
Prototypical application: digsig



Apr-25 Digital signatures 21

21

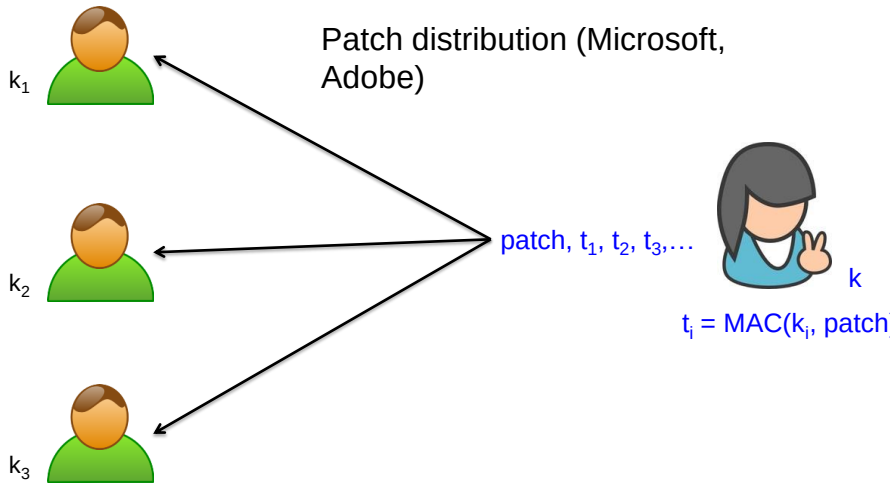
Prototypical application: MAC (1)



Apr-25 Digital signatures 22

22

Prototypical application: MAC (2)



Prototypical application: MAC

- 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!

Comparison to MAC →

- 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

%

25

Comparison to MAC →

- 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!

26

Digital signatures

THE RSA SIGNATURE SCHEME

27

Plain RSA

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

↳ We use public key for verification

Algorithm is exactly the same

28

Properties

- Computational aspects
  - *The same considerations as PKE*
- Security
  - Algorithmic attacks (Factoring)
  - Existential forgery (Subject to...)
  - Malleability

Apr-25 Digital signatures 29

29

2 types of forgeries: selective vs existential
1. Adversary controls the message: can obtain from message - dig. signature of message under his control.
2. Adversary can find message - dig. signature but has no control over message

Existential forgery

- Given public key (n, e), generate a valid signature for a random message x
  - Choose a signature σ
  - Compute $x = \sigma^e \bmod n$: if I do x^d I get σ. *
  - 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-25 Digital signatures 30

30

* $x^d \equiv (\sigma^e)^d \equiv \sigma^{e \cdot d} \equiv \sigma \bmod n$
↳ consistency of RSA

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 $\sigma_3 \equiv (\sigma_1 \times \sigma_2) \bmod n$ that is a valid signature of $x_3 \equiv (x_1 \times x_2) \bmod n$
    - PROOF. $x_3 = \sigma_3^e \equiv (\sigma_1 \times \sigma_2)^e \equiv \sigma_1^e \times \sigma_2^e \equiv x_1^{de} \cdot x_2^{ed} \equiv x_1 \cdot x_2 \bmod n$

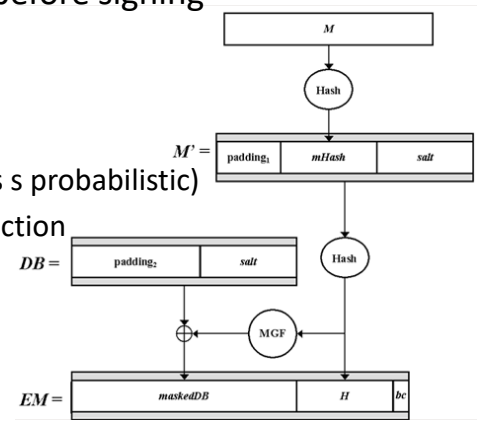
RSA Padding

- 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 PSS scheme specified in PKCS 1
    - Encoding Method for Signature with Appendix (EMSA)

You add redundancy and structure to the message

PSS

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



Digital Signatures

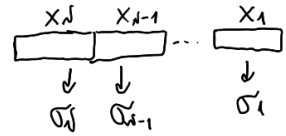
DIGITAL SIGNATURES VS HASH FUNCTIONS

Signing long messages

RSA requires x to be lower than modulo

- Consider RSA digsig
  - Message $0 \le x < n$ → small, few hundred bytes
    - E.g., n = 1024–3072 bits (128–384 bytes)
  - What if x > n?
  - An ECB-like approach is not recommended: divide X in blocks and sign them separately. ①
    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

35

① 

1. Perform signature multiple times
2. Signature is longer
3. But worse we have block reordering and substitution

Hash-and-Sign paradigm

- Given a signature scheme Σ = (G, S, V) for "short" messages of length n-bit
  - E.g. RSA
- Given a Hash function H: $\{0,1\}^* \rightarrow \{0,1\}^n$
- Construct a signature scheme Σ' = (G, S', V') for messages of any length
  - $\sigma = S'(privK, x) = S(privk, H(x))$
  - $V'(x, pubK, \sigma) = V(H(x), pubK, \sigma)$

36

Hash-and-sign paradigm

- THM. If Σ is secure and H is collision-resistant, then Σ' is secure
  - PROOF by contradiction FOR NEXT TIME
    1) Assume that the sender authenticates $x_1, x_2, \ldots$
    2) Assume the sender manages to forge (x', σ'), $x' \neq x_i$, for all i
    3) Let $h_i = H(x_i)$. Then, we have two cases
       1) If $H(x') = h_i$ for some i, then collision in H (contradiction)
       2) If $H(x') \neq h_i$, for all i, then forgery in Σ (contradiction)

Apr-25 Digital signatures 37

37

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-25 Digital signatures 38

38

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 x
    - Q.E.D

Dig sig vs hash properties

- 2nd preimage resistance
  - The protocol
    - Bob → Alice: x Sends alice an order, Alice returns it signed
    - 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 [←] claim that Alice has signed x' instead of x
    - Q.E.D

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

When hashes do not satisfy the three properties (especially last one), they are deprecated.

Apr-25 Digital signatures 41

41

Digital signatures

RSA-BASED BLIND SIGNATURES

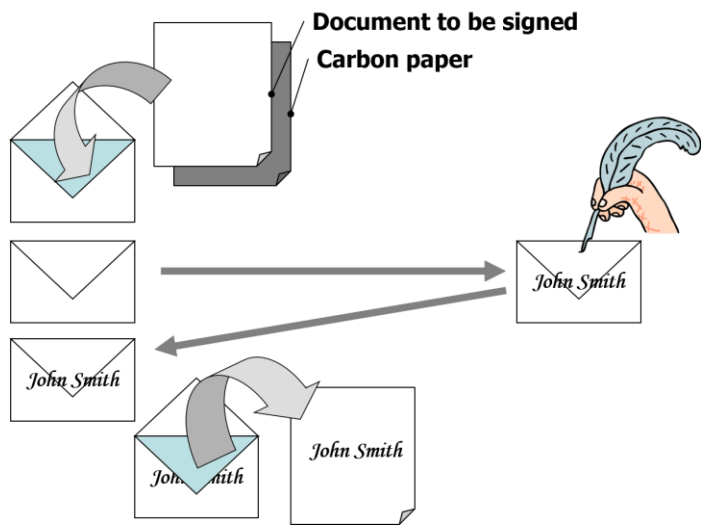
Apr-25 Digital signatures 42

42

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



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

45

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

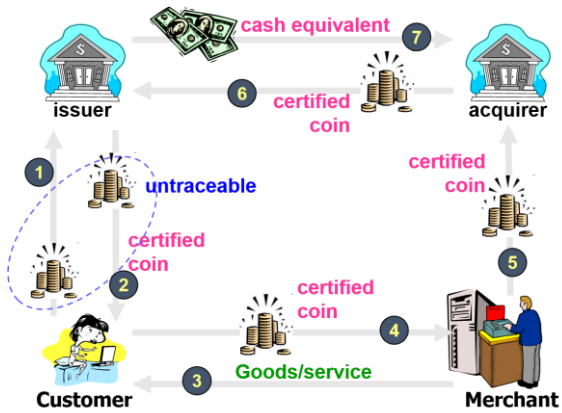
46

Applications

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

47

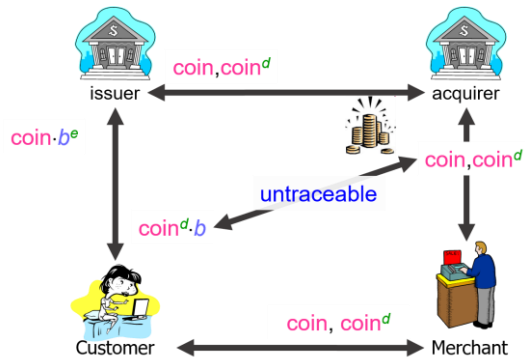
Digital cash



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

48

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

49

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

50

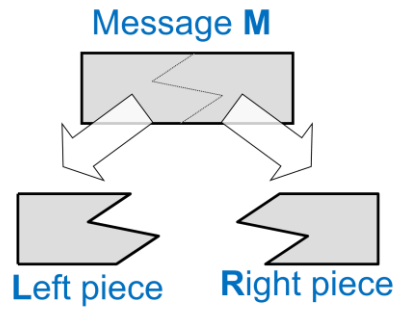
Double spending ↓

- 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

51

Secret splitting [→]

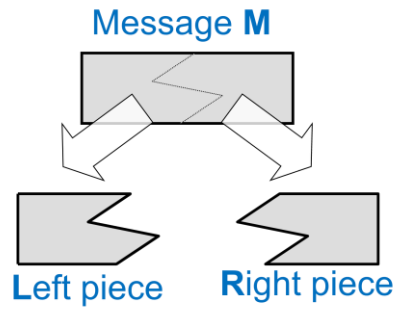
- Each piece alone gives no information on the message
- Both pieces make it possible to reconstruct the message



52

Secret splitting

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

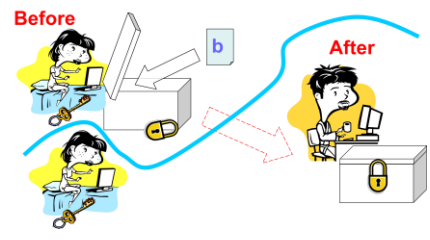


Apr-25 Digital signatures 53

53

Bit commitment [→]

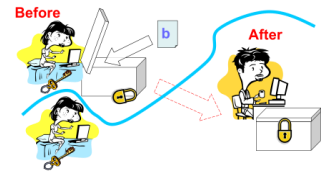
- Alice thinks of a number and Bob has to guess it.
- Alice thinks about the number but doesn't want to reveal it.
- Bob guesses the number but wants to be sure Alice doesn't change it.



Apr-25 Digital signatures 54

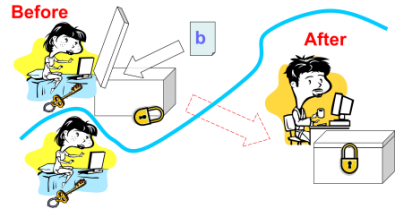
54

Bit Commitment [→]



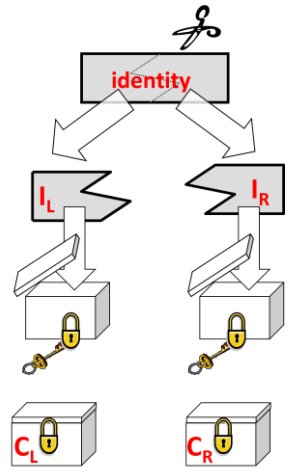
- Perfectly binding
  - It is theoretically impossible for Alice to alter her commitment after she makes it
- Perfectly concealing
  - It is theoretically impossible for Bob to find commitment without Alice revealing it
- THM There exists no commitment scheme which is both perfectly binding and perfectly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

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.

On solving double spending



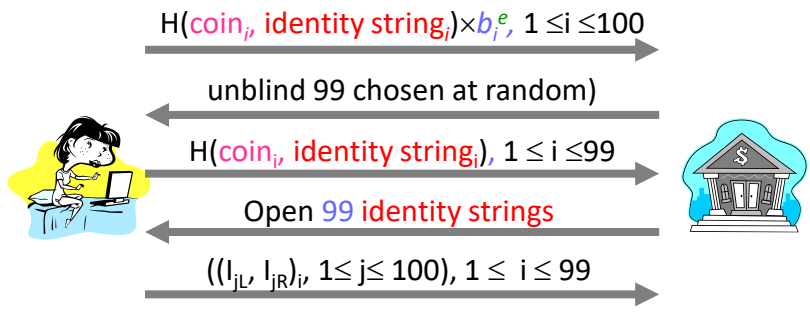
57

On solving double spending

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

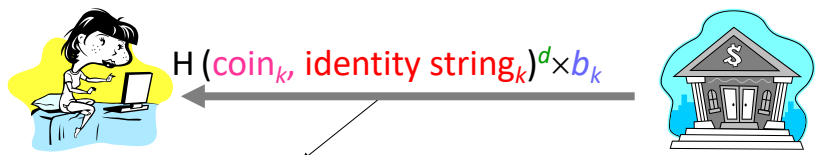
58

On solving double spending: cut-and-choose



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

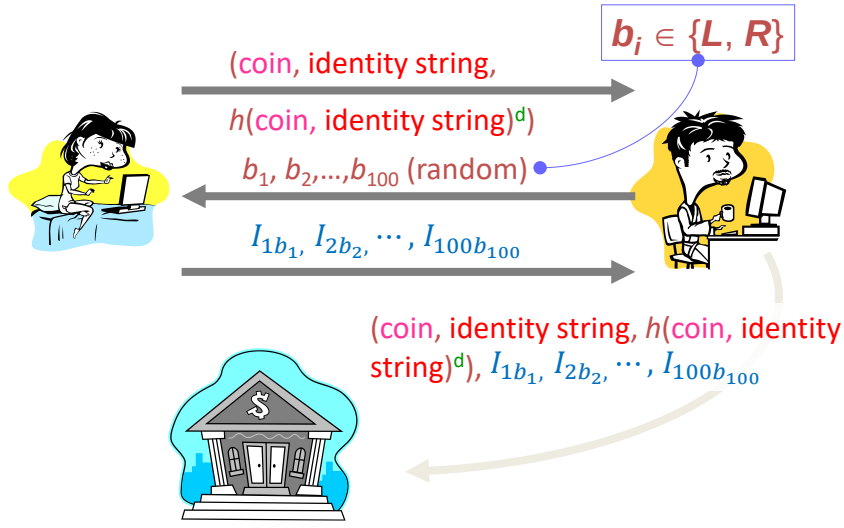
On solving double spending: withdraw



H(coin$_k$, identity string$_k$)^d Authenticated coin

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

On solving double spending: spend



61

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

62

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

63

Digital signatures

THE ELGAMAL SIGNATURE SCHEME

65

Elgamal in a nutshell

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

66

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

67

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-25 Digital signatures 68

68

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-25 Digital signatures 69

69

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 $s = (x - d \cdot r) k_E^{-1} \bmod p - 1$

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 $k_E^{-1} \bmod p$ by EEA (pre-computation)
- Signature verification
  - Two exponentiations by square-and-multiply
  - One multiplication

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 (➔)

72

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.

73

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!

74

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

75

Digital Signatures

DIGITAL SIGNATURE ALGORITHM (DSA)

76

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

77

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)

78

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

79

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

80

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

81

Proof [→]

- We show that a signature (r, s) satisfies the verification condition v ≡ r mod 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: $\alpha^{kE} \bmod p \equiv \alpha^{u1 + d\ u2} \bmod p$

[→]

Apr-25 Digital signatures 82

82

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 ≡ v mod q.

Apr-25 Digital signatures 83

83

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

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

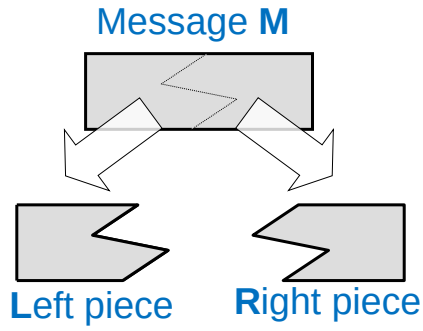
Computational aspects

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

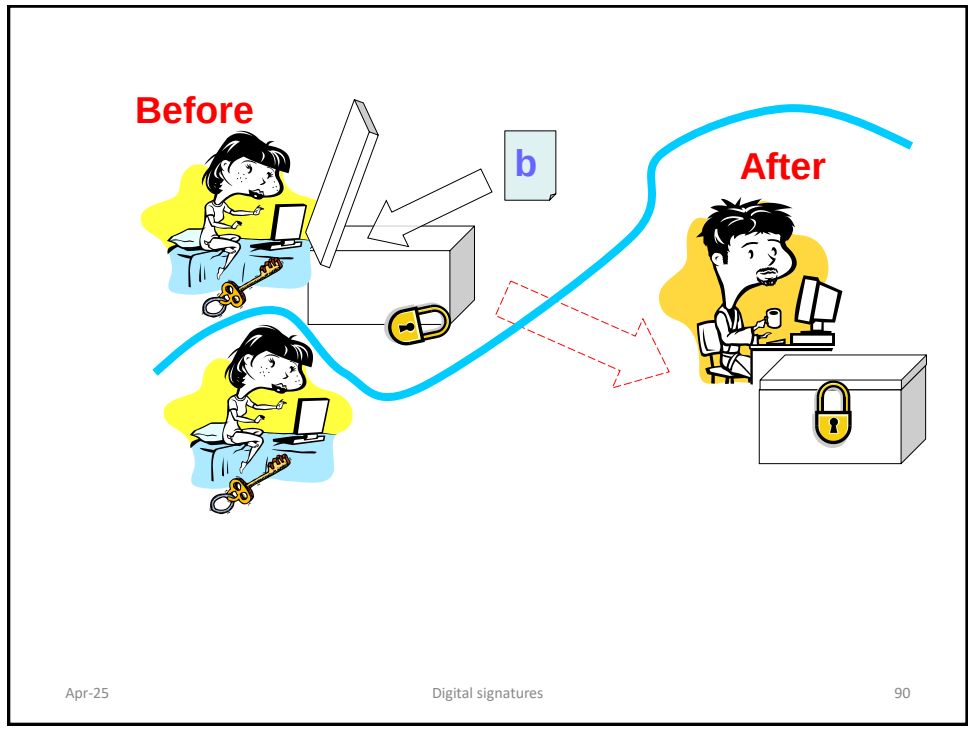
Security

- We have to protect from two different DLPs
  1. $d = \log_\alpha \beta \bmod p$.
     - 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

88



89



90