جامعة الشارقة

UNIVERSITY OF SHARJAH

Blind Digital Signatures

Dr. Qassim Nasir

Digital Signature – Why?

- To provide Authenticity, Integrity and Non -repudiation to electronic documents

- To use the Internet as the safe and secure medium for e-Governance and e-Commerce

- The originator of a message uses a signing key (Private Key) to sign the message and send the message and its digital signature to a recipient

- The recipient uses a verification key (Public Key) to verify the origin of the message and that it has not been tampered with while in transit

Digital Signature – What is Digital Signature ?

- Digital signatures are one of the most **important inventions** of modern secure data exchange.

- Digital signatures should be such that each user should be able to **verify signatures of other users**, but that should give him/her **no information how to sign** a message on behind of other users.

Digital Signature – What is Digital Signature ?

- The main **difference** from a **handwritten** signature is that **digital** signature of a message is intimately **connected** with the **message**, and for different messages is different, whereas the **handwritten** signature is **adjoined** to the **message** and always looks the **same**.

- Technically, digital signature is **performed** by a **signing** algorithm and it is **verified** by a **verification** algorithm

Digital Signature – Requirements Of Digital Signature.

- **Depending** on the **message** to be signed, the signature should be **a bit pattern**.

- The signature should use **some information unique** to the **sender** so that forgery and denial of service attack can be prevented.

- The digital signature should be **relatively easy** to **produce**.

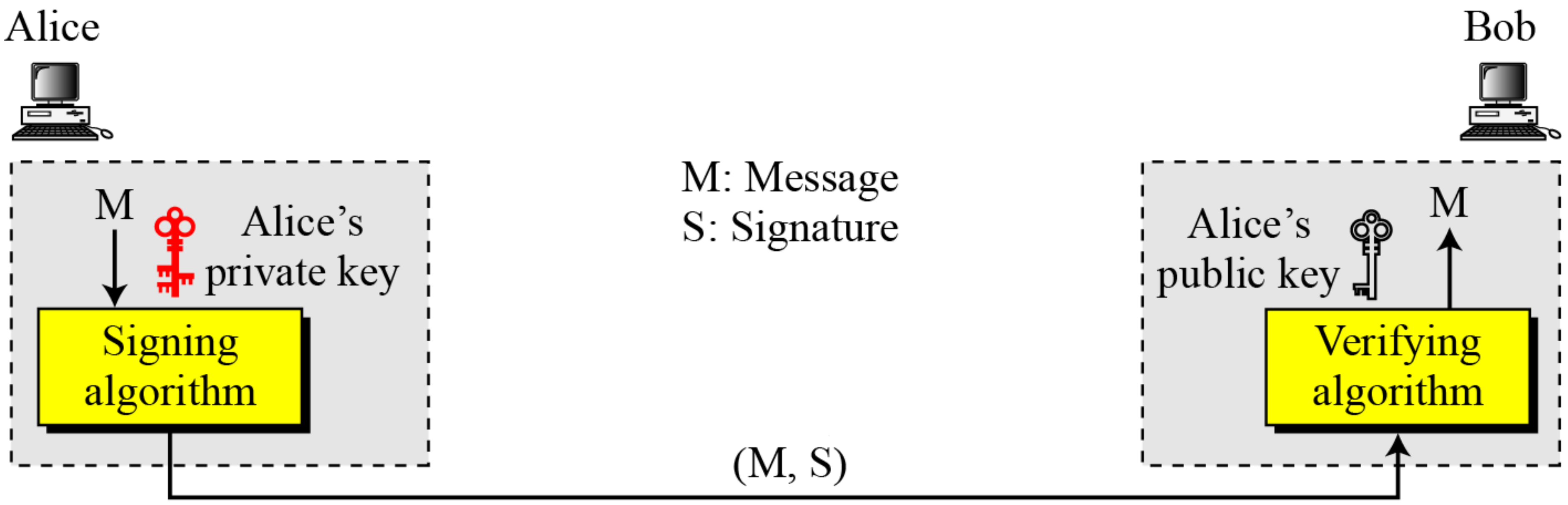
Digital Signature – Requirements Of Digital Signature.

- **Recognition** and **verification** of the digital signature should be relatively **easy**.

- **Forgery** of a digital signature should be **computationally** **infeasible**.

- Serve the purpose of
  - Authentication
  - Integrity
  - Nonrepudiation (by using a trusted party)

Digital Signature – What is Digital Signature ?



A digital signature needs a <u>public-key system</u>.
The <u>*signer*</u> signs with her <u>*private*</u> key; the <u>*verifier*</u> verifies with the signer's <u>*public*</u> <u>*key*</u>.

Digital Signature – What is Digital Signature ?

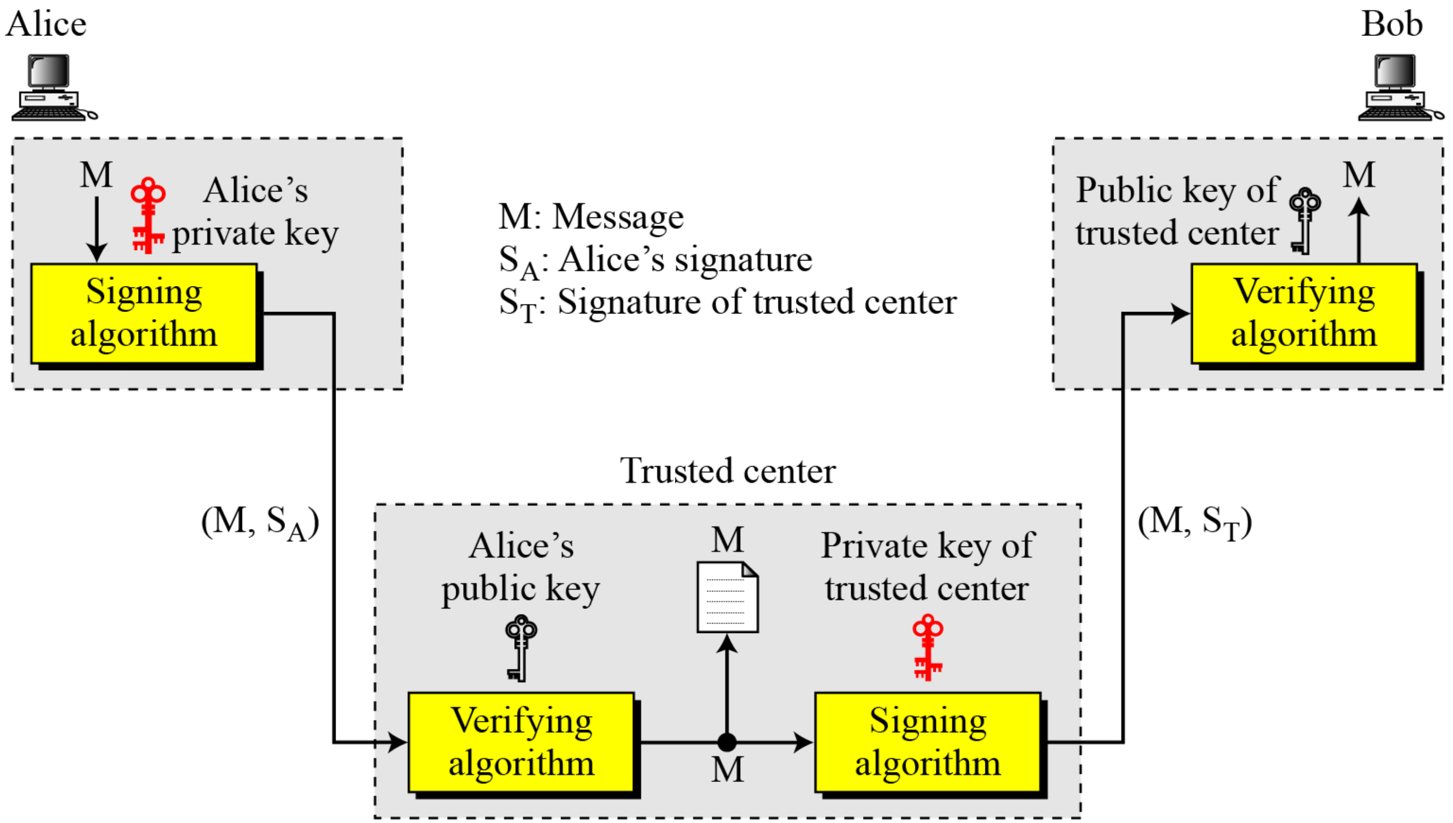
- A secure digital signature scheme, like a secure conventional signature can provide message authentication.

A digital signature provides message authentication.

- The ***integrity*** of the message is ***preserved*** even if we sign the whole message because we ***cannot get*** the ***same signature*** if the message is ***changed***

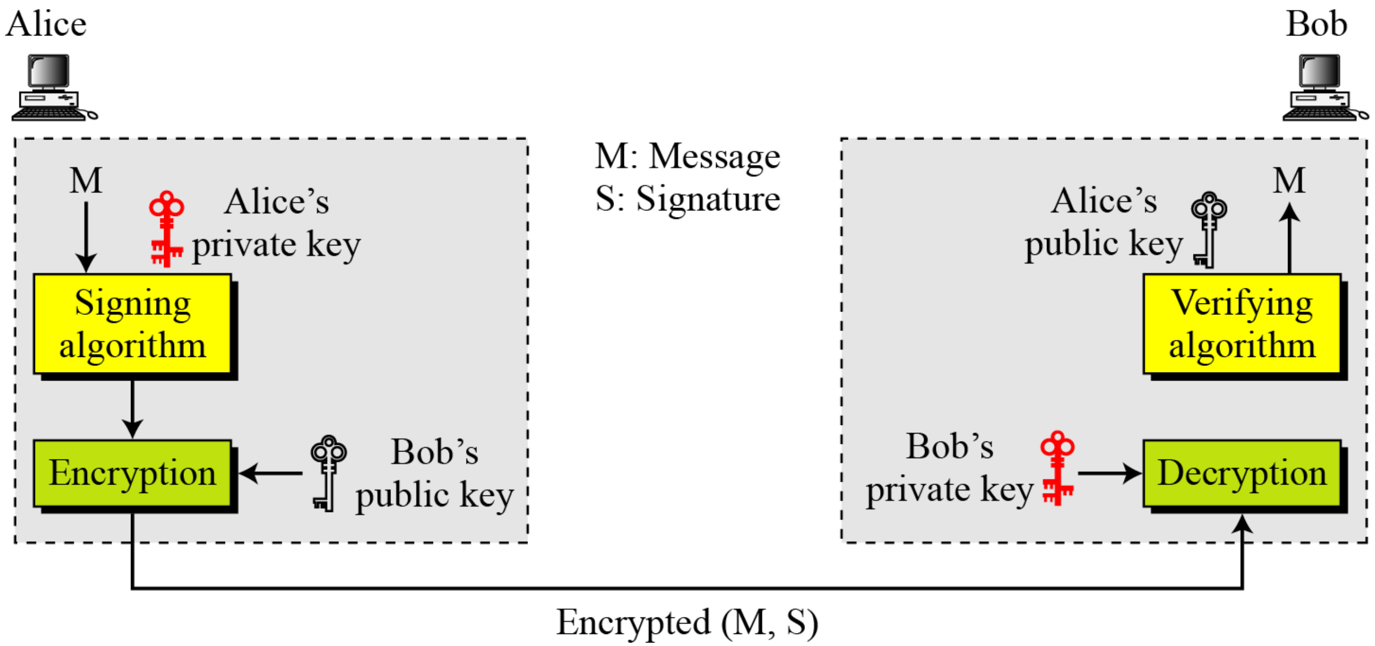
A digital signature provides message integrity.

Digital Signature – What is Digital Signature ?



Nonrepudiation can be provided using a trusted party.

Digital Signature – What is Digital Signature ?



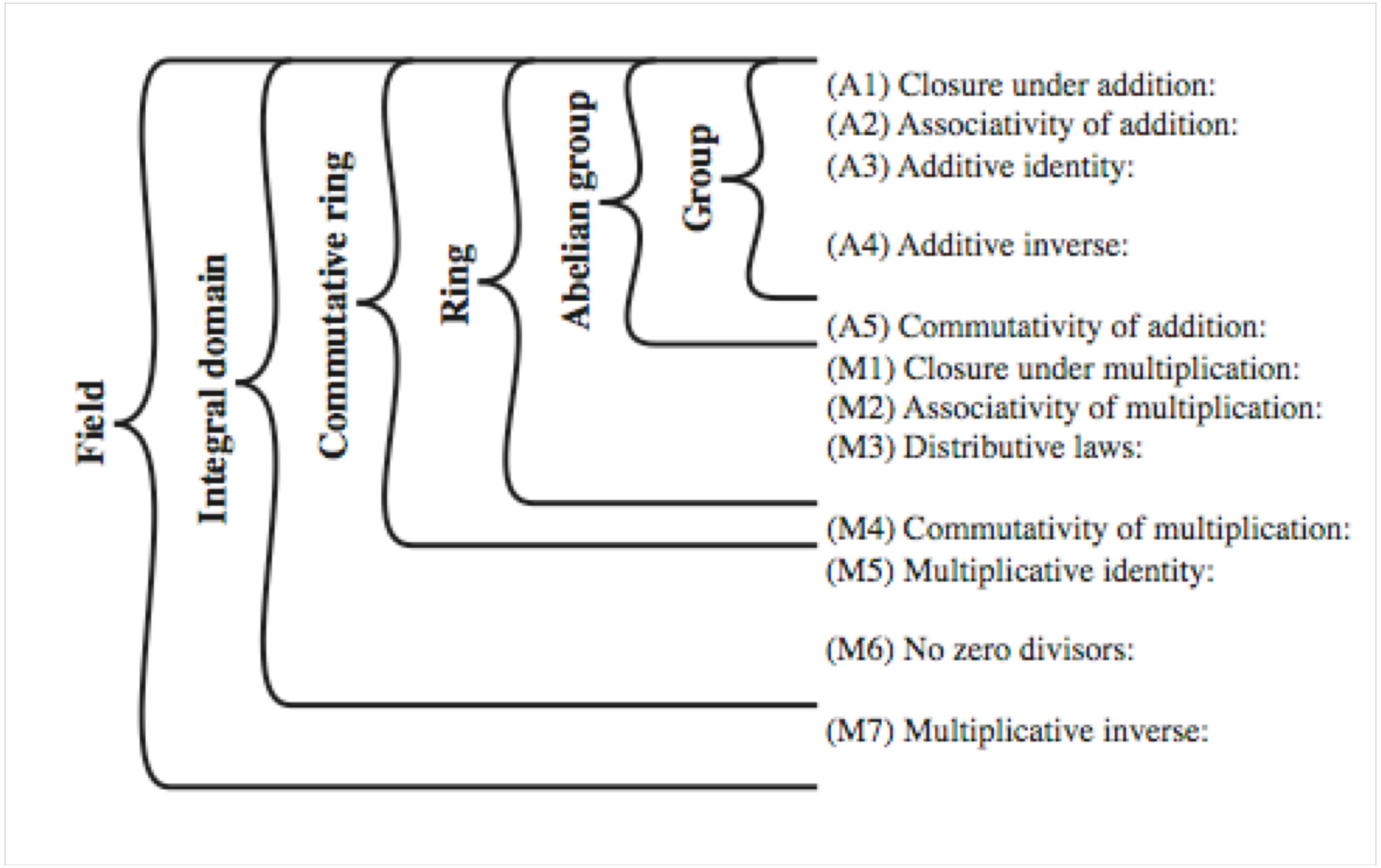
A digital signature *__does not provide privacy__*.
If there is *__a need for privacy__*, another layer of *__encryption/decryption__* must be applied.

Finite Fields

- Galois Field

- GF(p^n) = a set of integers {0, 1, 2, …, p^n - 1)
- where p is a prime, n is a positive integer

- It is denoted by {F, +, x}
- where + and x are the group operators



(A1) Closure under addition:
(A2) Associativity of addition:
(A3) Additive identity:
(A4) Additive inverse:
(A5) Commutativity of addition:
(M1) Closure under multiplication:
(M2) Associativity of multiplication:
(M3) Distributive laws:
(M4) Commutativity of multiplication:
(M5) Multiplicative identity:
(M6) No zero divisors:
(M7) Multiplicative inverse:

Finite Groups and Number Theory

- The formal definition of a group is that it should be a set, G, (e.g. the set of integers, Z) together with an operation, *, (usually called multiplication) which should be closed in G.

- That means that for any x, y € G, their combination, x*y, must also lie in G.

Finite Groups and Number Theory

- The operand, *, would also have to satisfy three axioms:

1. (Associativity). For all x, y, z € G, it follows that (x*y)*z = x*(y*z).
2. (Identity). There exists an element e € G such that x*e = x = e*x, for all x € G.
3. (Inverse). Each x € G has a (unique) inverse x^{-1} € G such that x^{-1}*x = e = x*x^{-1}.

- Since computers are bad at using infinite sets, a common approximation of the (positive) integers is: Zn = { 0, 1, 2, 3, ..., n-1 }

Finite Groups and Number Theory

- In abstract algebra, Zn is also an example of a finite group under addition modulo n.

- That means that all sums are reduced by multiples of n, such that only the remainder after a (integer) division by n is left.

- In this way, Zn becomes closed under the addition operator (mod n).

Finite Groups and Number Theory

- A more interesting group for cryptographic applications is:

Z_n* = { x € Z_n : gcd(x,n)=1 } under multiplication mod n.

- Zn * consists of all integers from Zn which are relatively prime to n, i.e. those that don’t share any common divisor with n (other than 1).

- In the case that n is a product of two (equally) large primes (n = p*q), then Zn* is said to be an RSA-group after the popular and well-known RSA crypto-system.

Factoring and Discrete Logarithm Problem

- Rivest-Shamir-Adleman (1977): RSA based on factoring.
  - Main idea: easy to find two large primes p and q, but very hard to find p and q from n = p . q.
  - RSA still most popular public key cryptosystem.

- ElGamal (1984): discrete logarithm problem (DLP).
  - Group G is set with operation (.) and each element has inverse.
  - Main idea: very easy to compute $h = g^x$ for given x, but very hard to find x given h and g.
  - Popular choices: finite fields and elliptic curves.

Discrete Logarithms

- Other common crypto-systems alternatively use the group Zp*, where p is a large prime such that p-1 doesn't have any small factors.

- Because p is prime, all numbers in Zp, except 0, will be relatively prime to p and hence:

  Zp* = { 1, 2, 3, ..., p-1 } Let g € Zp*.

- Fermat's little theorem then states that: $g^{p-1} \equiv 1 \pmod{p}$, for all g which are relatively prime to p.

Discrete Logarithms

- This means that the order of g is at most p-1 (can also be a factor of p-1). In the case that the order of g is p-1 (denoted by O(g) = p-1), then g becomes a generator of Zp*.

- That means that all elements in Zp* can be written as g^x (mod p), for some integer x

$$Zp^* = \langle g \rangle = \{ h : h = g^x \bmod p, x € Zp \} = \{ g, g^2, g^3, .., 1 = g^{p-1} \}$$

- The problem of finding a discrete logarithm is that, given h, g € Zp*, find an exponent x € Z such that: h = g^x (mod p), This is usually written: x = $\log_e$ h

Discrete Logarithms

- It is for this reason that exponentiation in prime groups is believed to be a one-way function, as is squaring in RSA-groups (the infeasible underlying problems are thus discrete logarithms and factoring).

- Discrete logarithms is used in Diffie-Hellman Key-Exchange, ElGamal Crypto-System and Schnorr- and DSS-Signatures.

Discrete Logarithms

- Some researchers have an intuitive feeling that discrete logarithms is a harder problem to solve than factoring, although the truth of this matter is better left unsaid.

- The scientific skills of solving these problems have often been improved in the same pace. When a great step in factoring is taken, the same advance is soon followed in computing discrete logarithms

Digital Signature Schemes

- Several digital signature schemes have evolved during the last few decades. Some of them have been implemented such as:

- RSA Digital Signature Scheme

- ElGamal Digital Signature Scheme

- Schnorr Digital Signature Scheme

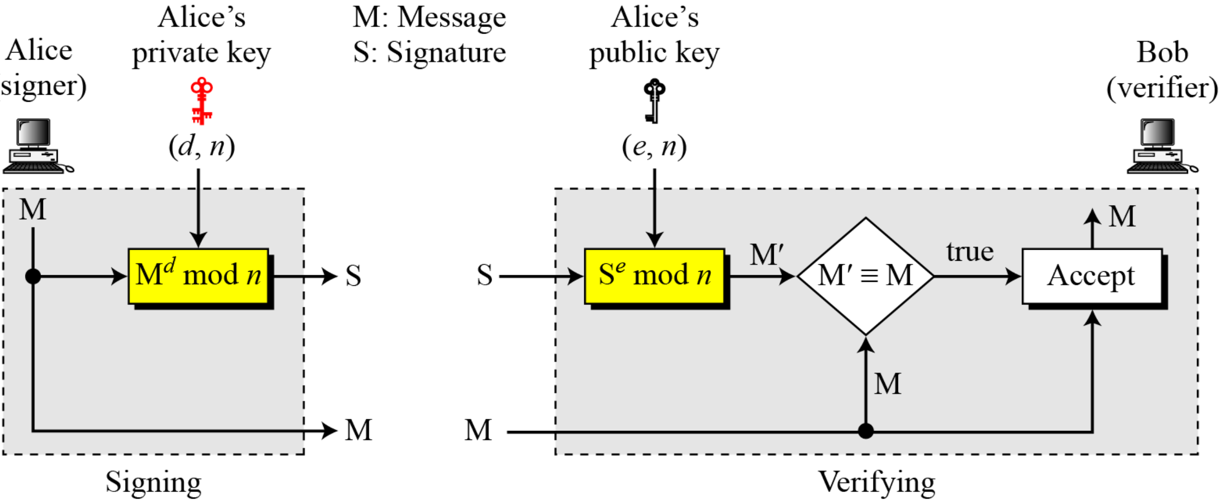
- Digital Signature Standard (DSS)

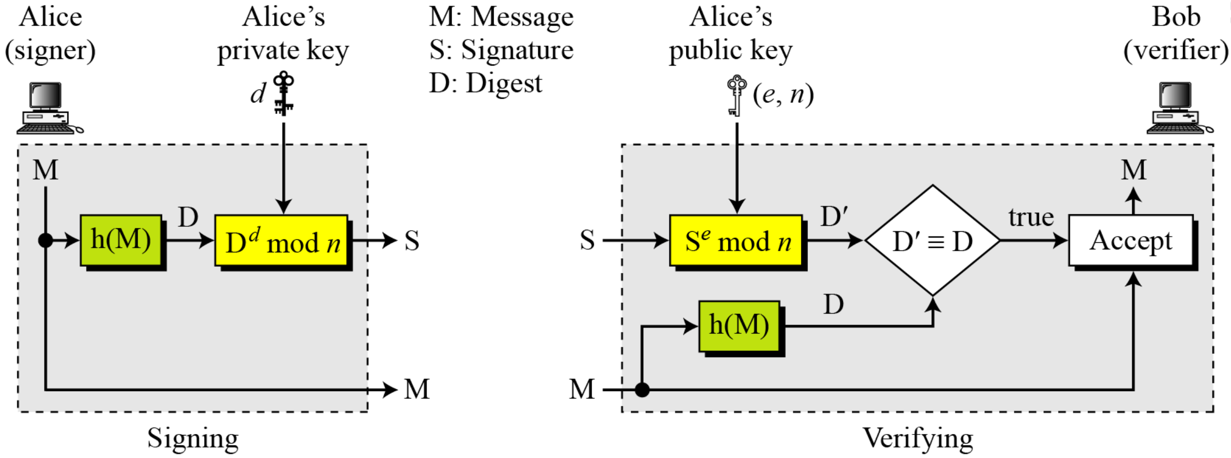
- Elliptic Curve Digital Signature Scheme

Digital Signature Schemes - RSA

- Key generation in the RSA digital signature scheme is exactly the same as key generation in the RSA
- In the RSA digital signature scheme, d is private; e and n are public.
- Signing and Verifying

RSA digital signature scheme



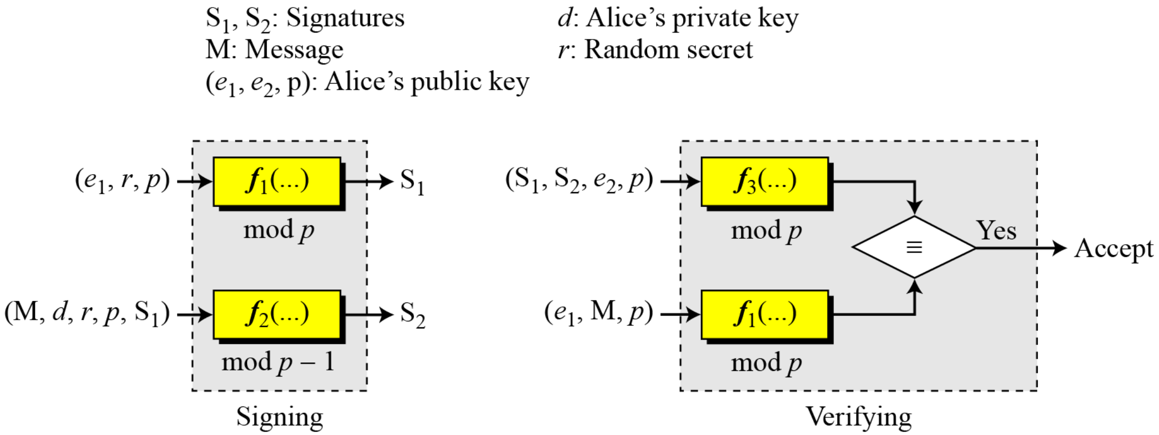


The RSA signature on the message digest

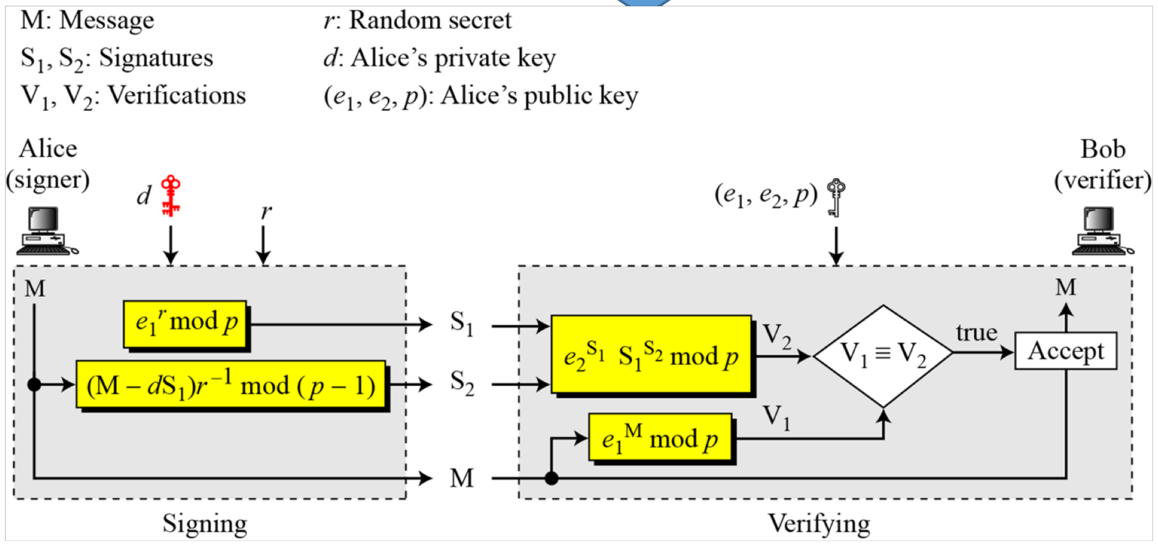
When the digest is signed instead of the message itself, the *susceptibility of the RSA digital signature* scheme *depends* on the *strength* of the *hash* algorithm

Digital Signature Schemes - ElGamal

- ➢Key generation in the ElGamal digital signature scheme is exactly the same as key generation in the Cryptosystem. Uses exponentiation in **a finite (Galois)**
- ➢In ElGamal digital signature scheme, **(e1, e2, p) is Alice's public key; d is her private key.**



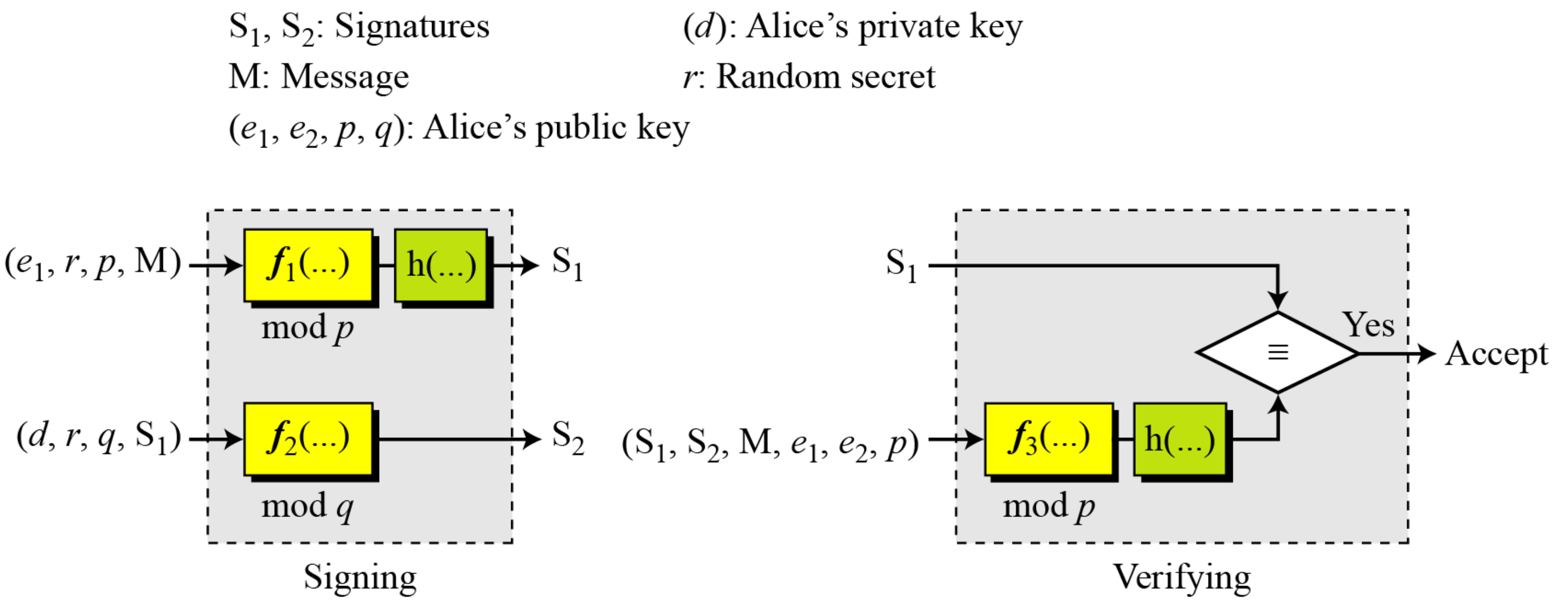
Verifying and Signing



Digital Signature Schemes - Schnorr

- ➢Also uses exponentiation in a finite (Galois)

- ➢Using a prime modulus p
  - ➢p–1 has a prime factor q of appropriate size
  - ➢typically p 1024-bit and q 160-bit (SHA-1 hash size)

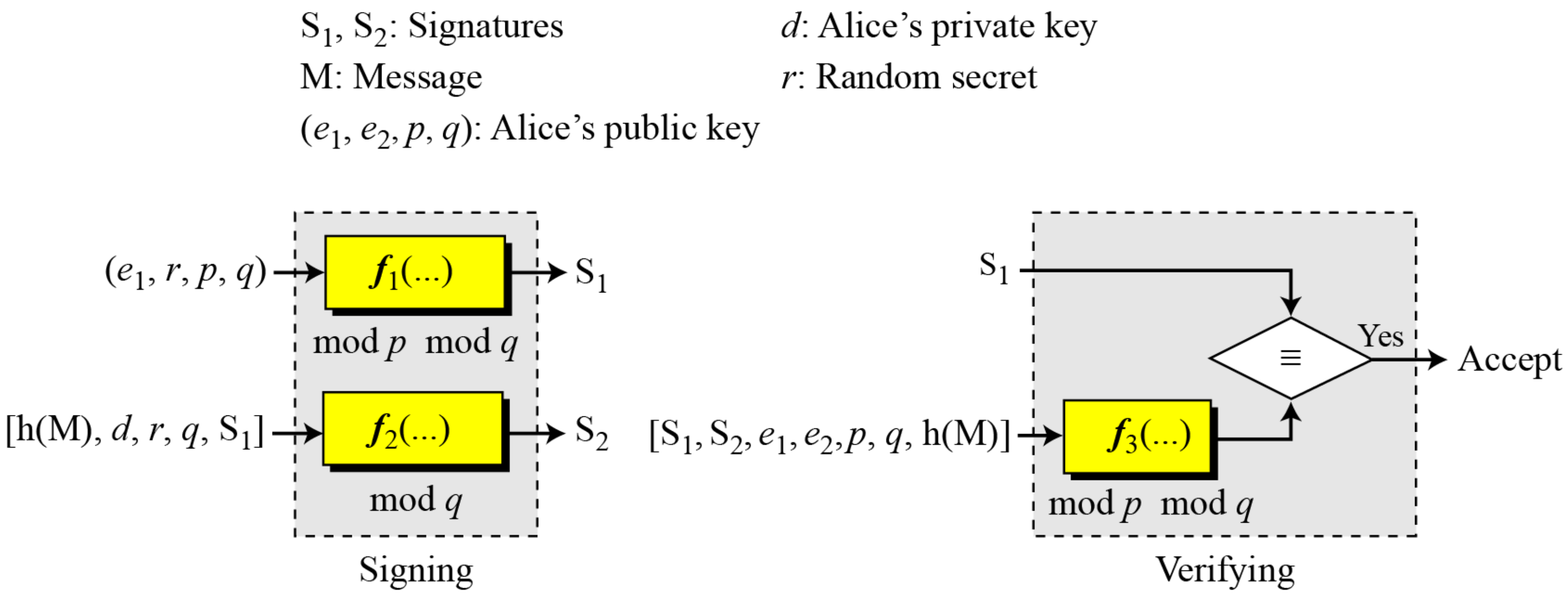


Digital Signature Standard (DSS)

- US Govt approved signature scheme
- Designed by NIST & NSA in early 90's
- Published as FIPS-186 in 1991
- Revised in 1993, 1996 & then 2000
- Uses the SHA hash algorithm
- DSS is the standard, DSA is the algorithm
- FIPS 186-2 (2000) includes alternative RSA & elliptic curve signature variants
- DSA is digital signature only

Digital Signature Standard (DSS)

- DSS Versus RSA
  - ***Computation*** of DSS signatures is ***faster*** than computation of **RSA** signatures when using the same p.
- DSS Versus ElGamal
  - DSS signatures are **smaller** than **ElGamal** signatures because **q is smaller than p.**



Elliptic Curve Digital Signature

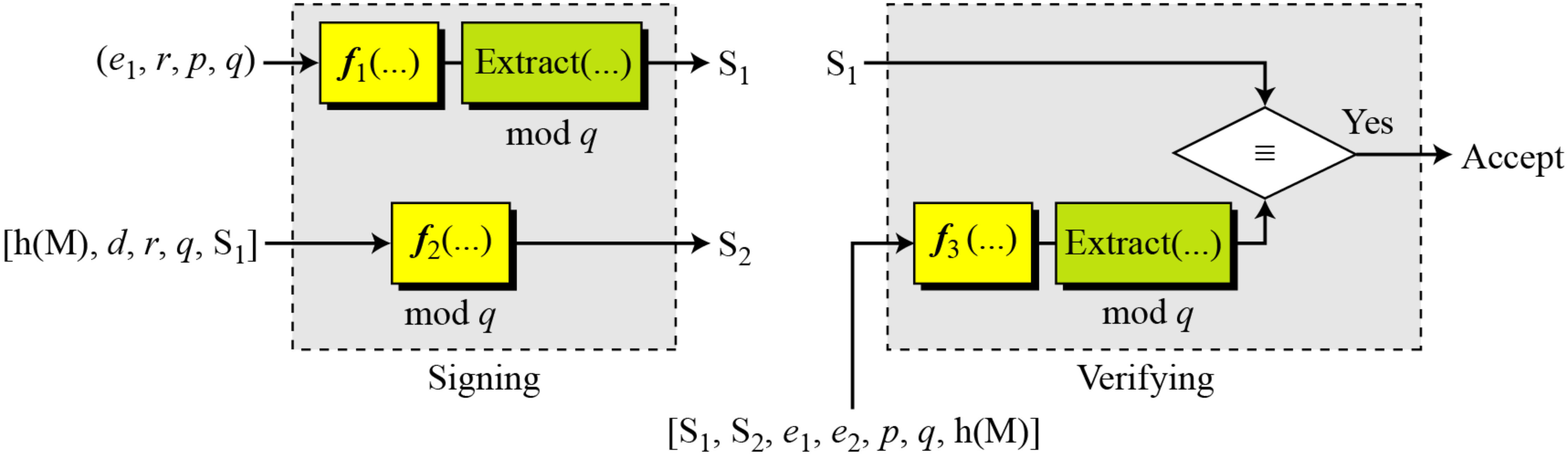
S_1, S_2: Signatures

M: Message

(a, b, p, q, e_1, e_2): Alice's public key

d: Alice's private key

r: Random secret



Why Elliptic Curve Cryptography (ECC)?

- Shorter Key Length
- Lesser Computational Complexity
- Low Power Requirement: Used in devices with limited storage and computational power such as wireless sensors devices, smart cards
- Web servers that need to handle many encryption sessions

Comparable Key Sizes for Equivalent Security

Symmetric Encryption (Key Size in bits)	RSA and Diffie-Hellman (modulus size in bits)	ECC Key Size in bits
56	512	112
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	512

What is Elliptic Curve Cryptography?

- Implementing Group Operations
  - Main operations - point addition and point multiplication
  - Adding two points that lie on an Elliptic Curve – results in a third point on the curve
  - Point multiplication is repeated addition
  - If P is a known point on the curve (aka Base point; part of domain parameters) and it is multiplied by a scalar k, Q=kP is the operation of adding P + P + P + P… +P (k times)
  - Q is the resulting public key and k is the private key in the public-private key pair

What is Elliptic Curve Cryptography?

- Discrete Log Problem (DLP)
  - The security of ECC is due the intractability or difficulty of solving the inverse operation of finding k given Q and P
  - This is termed as the discrete log problem
  - Methods to solve include brute force and Pollard’s Rho attack both of which are computationally expensive or unfeasible
  - The version applicable in ECC is called the Elliptic Curve Discrete Log Problem
  - Exponential running time

Elliptic Curve Schemes

- Elliptic Curve Digital Signature Algorithm (ECDSA)
- Elliptic Curve Pintsov Vanstone Signature(ECPVS)
- Elliptic Curve Diffie-Hellman (ECDH)

Elliptic Curve Digital Signature Algorithm (ECDSA)

➢Signature Generation

Once we have the domain parameters and have decided on the keys to be used, the signature is generated by the following steps.

1. A random number k, $1 \leq k \leq n-1$ is chosen
2. $kG = (x_1,y_1)$ is computed. x_1 is converted to its corresponding integer x_1'
3. Next, $r = x_1 \bmod n$ is computed
4. We then compute $k^{-1} \bmod q$
5. e = HASH(m) where m is the message to be signed
6. $s = k^{-1}(e + dr) \bmod n$, where d is the private key of the sender.

We have the signature as (r,s)

Elliptic Curve Pintsov Vanstone Signature (ECPVS)

- Signature scheme using Elliptic Curves
- More efficient than RSA as overhead is extremely low

Elliptic Curve Pintsov Vanstone Signature ECPVS

- **Signature Generation:** The plaintext message is split into two parts: part C representing the data elements requiring confidentiality and part V representing the data elements presented in plaintext. Both the parts are signed. The signature is generated as follows:

1. A random number k, $1 \leq k \leq n-1$ is chosen.
2. Calculate the point R on the curve (R = kG).
3. Use point R and a symmetric encryption algorithm to get $e = T_R(C)$.
4. Calculate a variable d such that $d = HASH(e \,||\, I_A \,||\, V)$, where I_A is the identity of the mailer terminal.
5. Now calculate the other part of the signature s as follows: $s = ad + k(\text{mod } n)$; where a is the private key of the sender.

The signature pair (s,e) is transmitted together with the portion V of the plaintext.

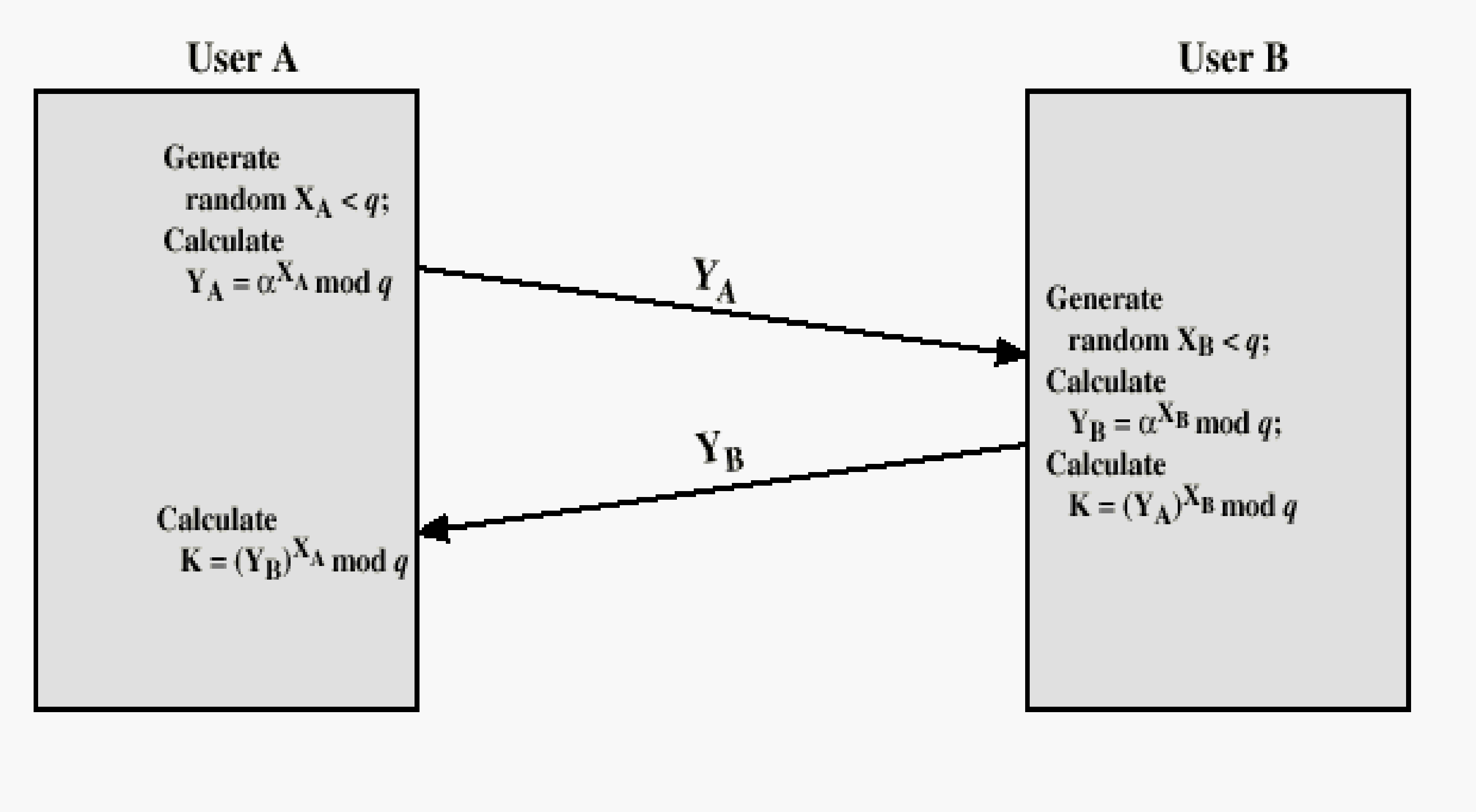
Elliptic Curve Diffie-Hellman (ECDH)

- Elliptic curve variant of the key exchange Diffie-Hellman protocol.

- Decide on domain parameters and come up with a Public/Private key pair

- To obtain the private key, the attacker needs to solve the discrete log problem

Elliptic Curve Diffie-Hellman ECDH



User A

Generate random $X_A < q$;
Calculate $Y_A = \alpha^{X_A} \bmod q$

Calculate $K = (Y_B)^{X_A} \bmod q$

User B

Generate random $X_B < q$;
Calculate $Y_B = \alpha^{X_B} \bmod q$;
Calculate $K = (Y_A)^{X_B} \bmod q$

Summary and Variations

Summary

- Digital signature depends upon the message and some information unique to the signer to prevent forgery and denial. Anyone should be able to verify.

- ElGamal/Schnorr/DSA signatures use a per-message secret key and are based on exponentiation

Variations

- 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.

Blind signature protocol

- Introduced by David Chaum in 1983. It’s a problem ***signing documents*** you ***can’t read***. Blind signatures are only used in special situations
- David Chaum Born 1955 is an American computer scientist and cryptographer.
  - He is famous for developing ecash, an electronic cash application that aims to preserve a user’s anonymity.
  - He has also invented many cryptographic protocols and founded DigiCash, an electronic money corporation
  - His 1981 paper, "Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms", laid the groundwork for the field of anonymous communications research
- Blind signatures are typically employed in privacy-related protocols where the signer and message author are different parties. Examples include cryptographic election systems and digital cash schemes. .

Blind signature protocol

- David Chaum's and at the time , he was pursuing his doctorate in computer science at the University of California, Berkeley, and he was devising “cryptographic protocols for establishing trust between mutually untrusting parties.”

- David Chaum's pioneering work aimed at creating an electronic version of money. To achieve this goal, he introduced the notions of "coins“ and "blind signatures". He claimed that this was the only way to ensure the required anonymity: in real life, a coin cannot be easily traced from the bank to the shop, furthermore, two spendings of a same user cannot be linked together.

Blind signature protocol

- These are two privacy properties of real coins that Chaum wanted to mimic: untraceability and unlinkability.

- He proposed an electronic coin to be a number with a certificate (a signature) blindly produced by the Bank; it is withdrawn from the Bank, spent by the user and deposited by the shop. Therefore, all the security of the system relies on the security of the blind signature that we use.

Blind signature protocol

- In all the proposed electronic cash schemes, two main blind signature schemes have been used.

- The first was introduced by Chaum and is a transformation of the RSA signature scheme . And more recently, Brands presented a new scheme derived from the Schnorr signature scheme .

Blind signature protocol

- Blind signature schemes must satisfy the following requirements

- **Anonymity:** also known as unlinkability, which prevents the signer from linking later the blinded message to unblinded version that it may be called upon to verify.

- **Verifiability:** The receiver of the signature should be able to verify the stripped signature was formed using signer's private key

Blind signature protocol

- Blind signatures are typically employed in privacy-related protocols where the **signer** and **message author** are ***different parties***.

- Blind signature schemes see a ***great deal of use*** in applications where ***sender privacy*** is important.

- blind signature protocol has two properties, ***blindness*** and ***intractableness*** (difficult to manipulate).

Blind signature protocol

- blind signature schemes are based on a trapdoor function such as the integer factorization problems(IFP), discrete logarithm problems(DLP).

- But most of the schemes fail to meet the above two fundamental properties

- Therefore, most of researchers aim to design a blind signature scheme that possesses both the above properties

Blind signature protocol -Terms

- Literally: anonymous = “without a name”

- Recall: Bitcoin/blockchain addresses are public key hashes rather than real identities. Computer scientists call this pseudonymity

- anonymity = pseudonymity + unlinkability = “Different interactions of the same user with the system should not be linkable to each other”,

- **Payer anonymity:** the identity of the payer should remain unknown; the payer must not be identifiable.

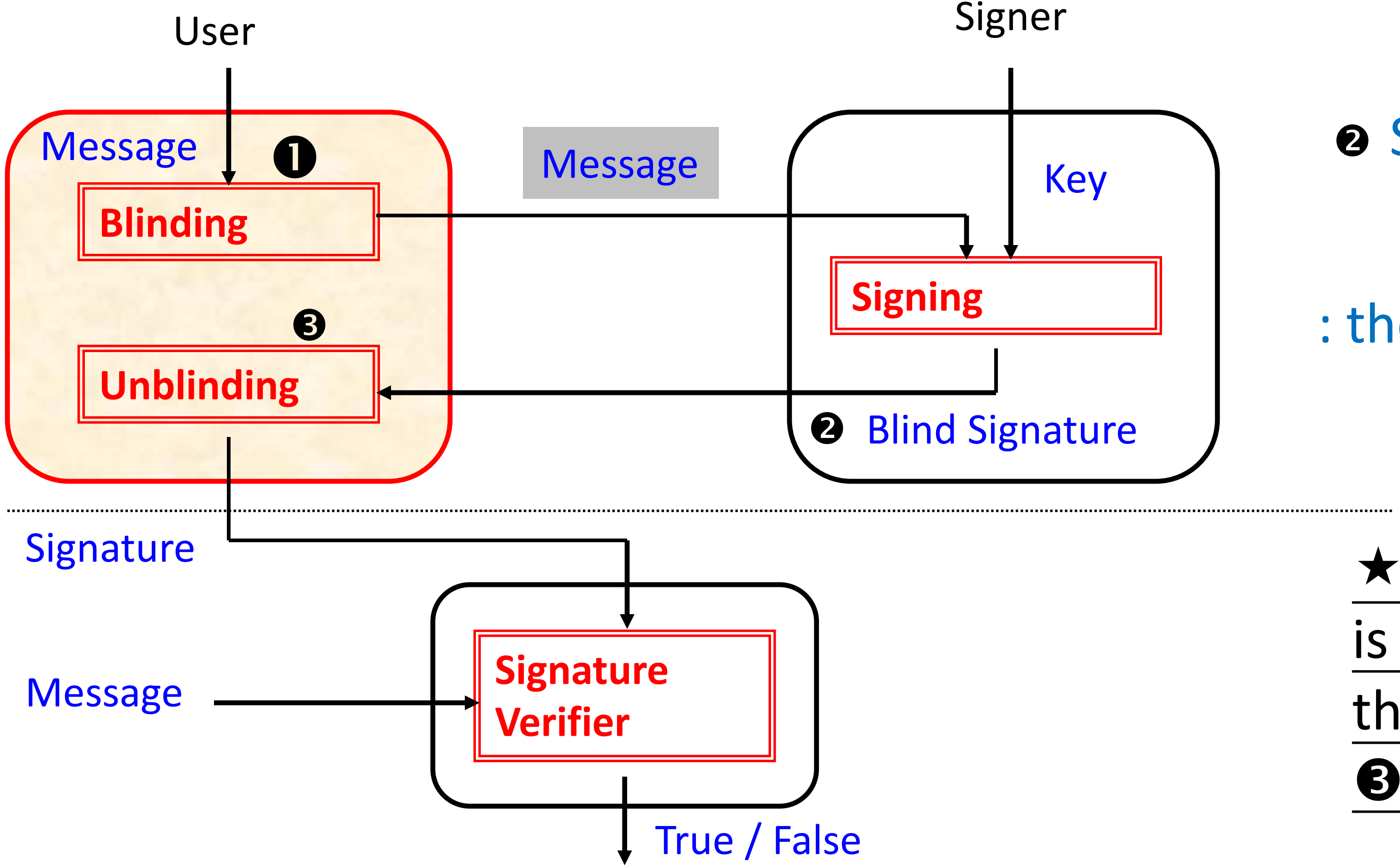
Blind signature protocol -Terms

- **Payer untraceability:** inability to trace a payment back to payer.
- **Payment untraceability:** inability to trace a payer to obtain information about the payments he/she has performed
- **Payment unlinkability:** inability to link different payments, as made by the same payer. In some systems payer the payment unlinkability is not needed.
- Anonymous e-Cash: History: Proposed by David Chaum in 1982, Based on Blind Signatures: Two-party protocol to create digital signature without signer knowing what she signs

Signature Generation and Verification

➊ “Message” : the **blinded** message

❸ Signature on “Message”: to be obtained after **unblinding**



❷ Signature on “Message” : the **blind** signature

★ *Unlinkability*: it is intractable for the signer to link ❸ to ❷

The Protocol

User

M : the underlying set of messages

R : a finite set of random integers

Signer

$m \in M$

$r \in R$

Publish V

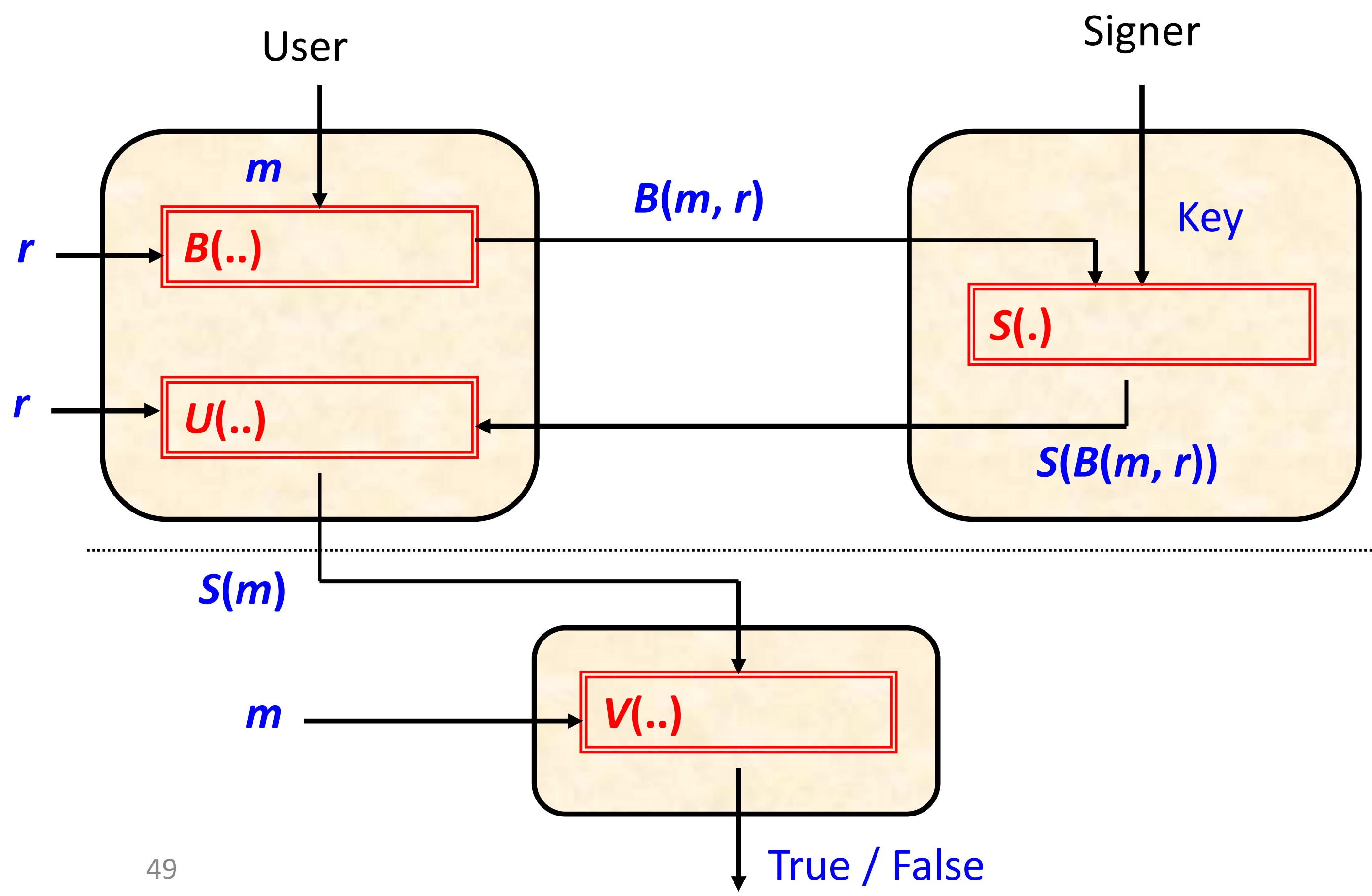
$B(m, r)$ →

← $S(B(m, r))$

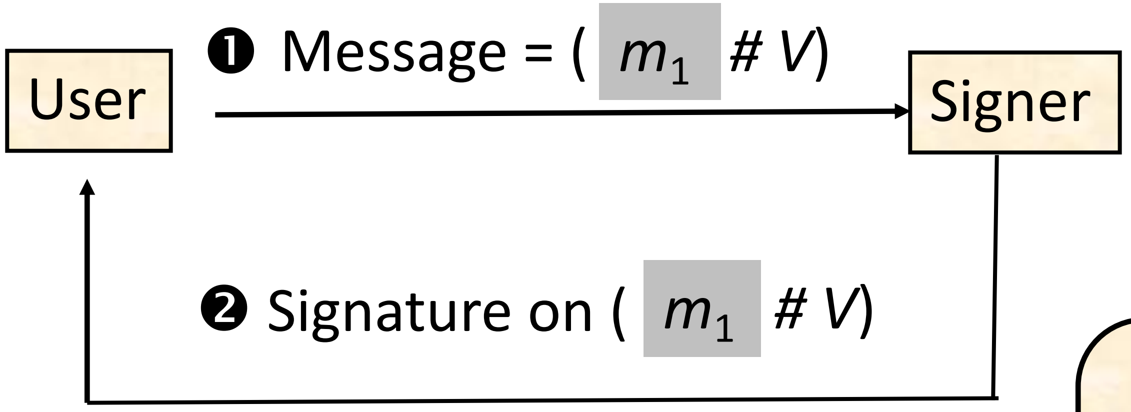
$U(S(B(m,r)), r) = S(m)$

Signature-message pair: ((*S*(*m*), *m*))

$V(S(m), m) = \text{True}$



Partially Blind Signatures



User

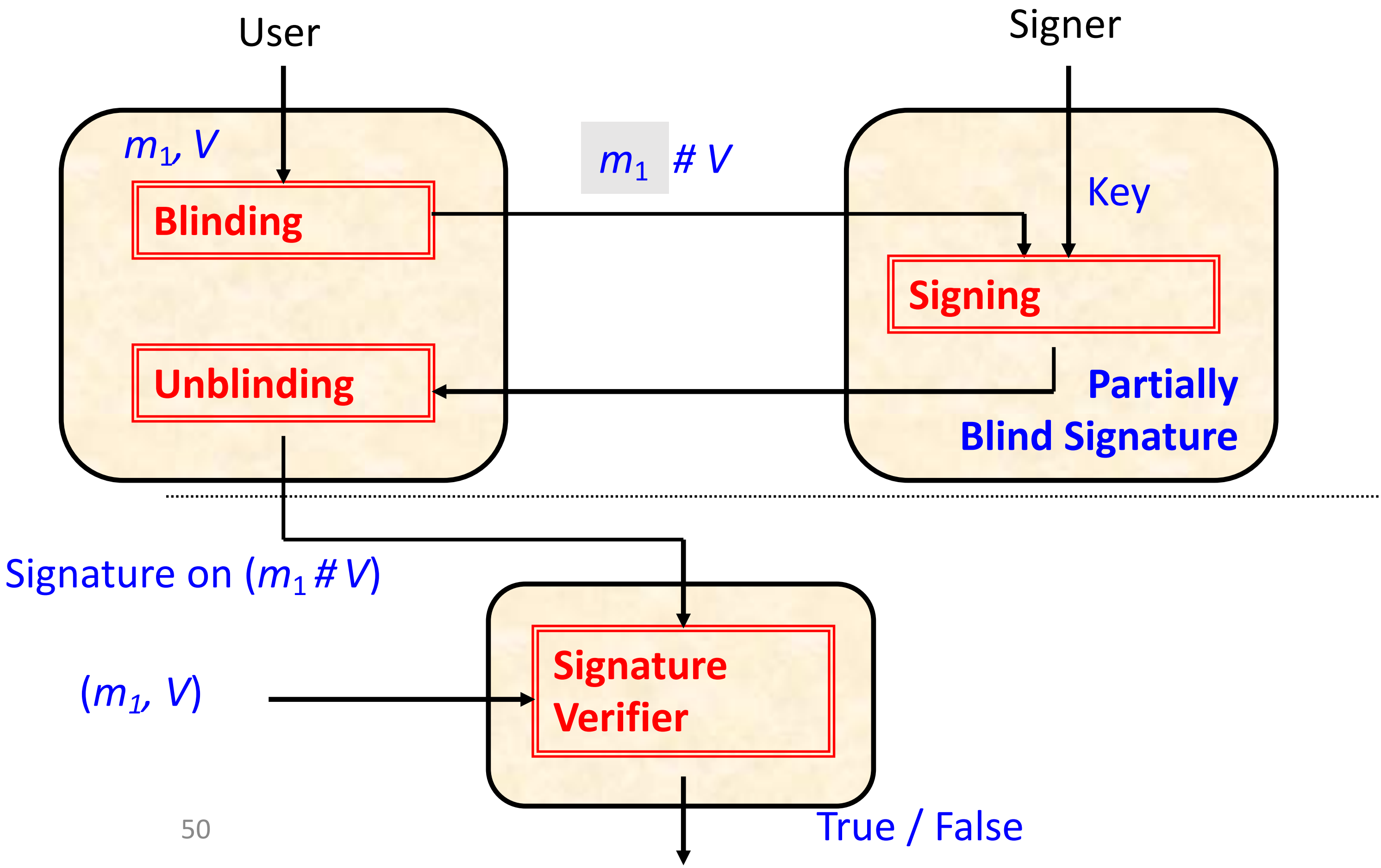
❶ Message = (m_1 # V)

Signer

❷ Signature on (m_1 # V)

❸ The signer's signature on (m_1 # V)

★ **All of the signatures with the same V are indistinguishable from the signer's point of view.**



User

m_1, V

Blinding

Unblinding

m_1 # V

Signer

Key

Signing

Partially Blind Signature

Signature on (m_1 # V)

(m_1, V)

Signature Verifier

True / False

Partially Blind Signatures- The Protocol

User

Signer

$M_1 \in M$

$r \in R$

V *expiration date*

Publish V

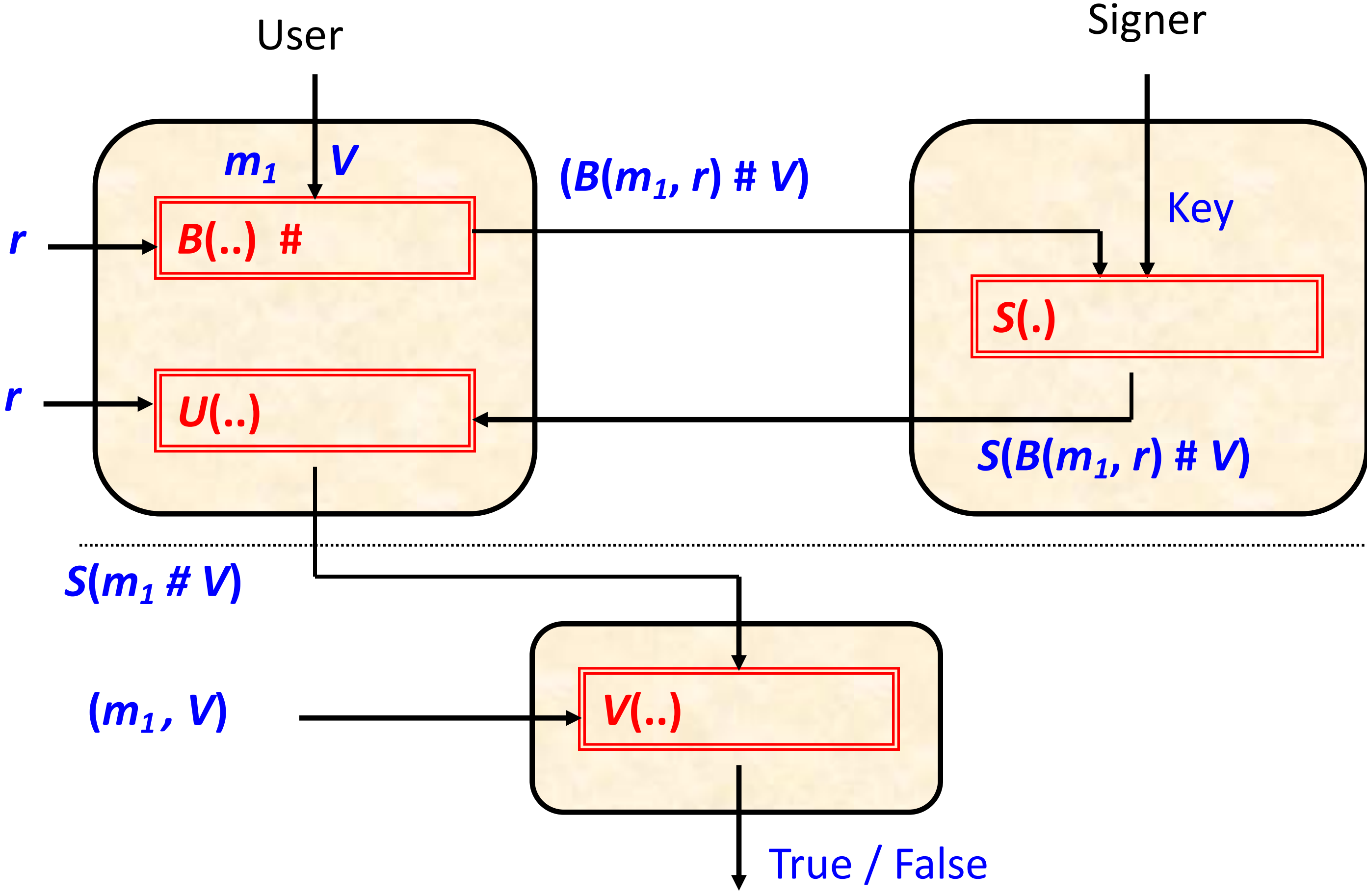
$(B(m_1, r) \# V)$ →

$S(B(m_1, r) \# V)$ ←

$U(S(B(m_1, r) \# V), r) = S(m_1 \# V)$

Signature-message triple: $(S(m_1 \# V), m_1, V)$

$V(S(m_1 \# V), (m_1 \# V)) = \text{True}$



User

Signer

m_1

V

r

$B(..)\ \#$

$(B(m_1, r) \# V)$

Key

$S(.)$

r

$U(..)$

$S(B(m_1, r) \# V)$

$S(m_1 \# V)$

(m_1, V)

$V(..)$

True / False

Blind signature protocol –Blind RSA

- **Initializing:** Based on RSA public key cryptosystem,

- the bank randomly chooses two large prime number p and q, and computes n = p·q and ϕ (n) = (p-1)(q-1).

- It then determines a pair of public and private keys(e,d), satisfying e·d ≡ 1 (mod ϕ(n)) with gcd(e, ϕ (n)) = 1, and both e and d less than ϕ (n).

- the bank publishes (n, e, f), which f is an appropriate public exponent generation function, f(v) must be different for different value of v; where v is predefine message that contains expiration date of the e-cash.

Blind signature protocol –Blind RSA

- **Withdrawing:**

- If a customer decides to withdraw e-cash from the bank,
  - he/she randomly chooses two integers m and r in Z* , where m is a message and r is a blind factor; r is relatively prim to n and
  - computes $\alpha \equiv (r^{ev}. m \bmod n)$ where v is a message predefined by the bank and contains an expiration date of the e-cash.
- The customer then sends α and v to the bank.
  - After receiving (α, v), the bank first verifies whether or not v is correct.
  - If it is correct, the bank sends $s' \equiv (r^{(ev)}\text{-}1 \bmod n)$ to the customer and deducts true coins from customer's account in the bank.

Blind signature protocol –Blind RSA

- **Unblinding:** After receiving s', the customer computes $s \equiv (r^{-1}s' \bmod n)$ and gets his/her e-cash (m, s, v).

- **Depositing:** When the customer uses the e-cash,
- the payee first verifies v is correct and $s^{ev} \equiv m \bmod n$.
- If they are correct, he/she calls the bank to check whether the e-cash has been already spent, i.e. double spending checking.
- If the e-cash has not been spent, the payee accepts the payment and deposits the e-cash into his/her account, and bank stores (m, s, v) in its database for double-spending and adds money to the payee's account.

Blind signature protocol –Blind Schnorr

- Some examples of cryptographic primitives that uses the discrete logarithm problem instead of factoring to construct one-way functions

- C. P. Schnorr improved the efficiency of Chaum protocol in order to for it to be implementable on smart cards

- He suggested that the arithmetic should be performed in the subgroup Gq and also that the challenge c possibly could be picked from the interval {0, ..., 2^t - 1}, for a security parameter t. Schnorr used t = 72 in his paper.

- Apart from that, Schnorr's protocol is practically identical to Chaum's version.

Blind signature protocol –Schnorr

- In this protocol, Alice decide the challenge to the Bank (the signature issuer) from a hash on the (blind) message she wishes to get signed.

- When the protocol is finished, Alice has received a non-interactive signature on the message of her choice.

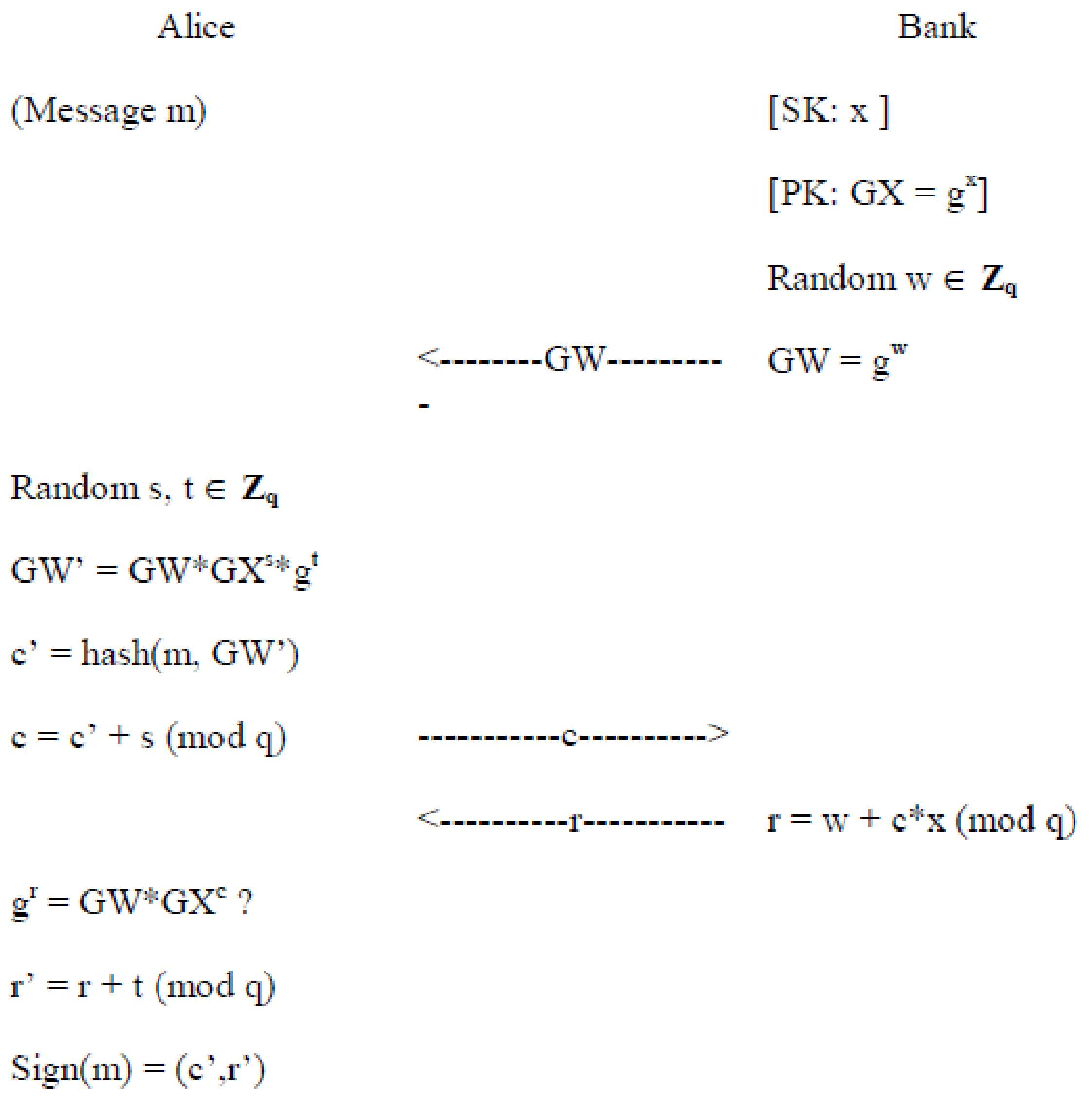
- The Bank on the other hand signs with an interactive signature scheme but is not concerned with what it actually signs.

Blind signature protocol –Schnorr

- It can be viewed as a reversed authentication-scheme, where the Bank identifies itself to Alice while she uses the received data to calculate a (Bank-) signature on her secret message.

- This clearly shows that one should be careful not to mix authentication- and signature-schemes, since it’s easy to unnoticedly sign messages while identifying oneself to somebody.

Blind signature protocol –Schnorr



Alice		Bank
(Message m)		[SK: x]
		[PK: GX = g^x]
		Random w ∈ $\mathbf{Z}_q$
	<--------GW--------- -	GW = g^w
Random s, t ∈ $\mathbf{Z}_q$		
GW' = GW*GX^s*g^t		
c' = hash(m, GW')		
c = c' + s (mod q)	-----------c---------->	
	<----------r-----------	r = w + c*x (mod q)
g^r = GW*GX^c ?		
r' = r + t (mod q)		
Sign(m) = (c',r')		

Blind signature protocol –Schnorr

- Why does the above protocol work? Well, since the Bank never sees the message it signs, nor the final signature (c’, r’), is it clear that the signing protocol is a blind signature issuing protocol.

- By verifying that the relation c’ = hash(m, $g^{r'}$ / GXc’) holds it can be seen that (c’, r’) is actually a Schnorr signature on m.

- That this is the case becomes clear by checking that:

- $g^{r'} = g^{r+t} = g^{w+c*x}\, g^{t} = GW'*GX^{c'}$

- and thus GW’ = $g^{r'}/GX^{c'}$, which is what Bob has to compute before he verifies the signature by checking that c’ = hash(m, GW’).

Blind signature protocol –Schnorr

- Although the above blind signature issuing protocol makes it possible to get blind Bank signatures on freely chosen messages, is it not possible however to receive such signatures without the interaction with the Bank, i.e. it is not possible for Alice to forge the Bank's signature all by herself.

- For this reason it would actually be possible to use the above protocol in designing an anonymous on-line payment system, such as David Chaum's ecash.

- The Bank would then have to keep all the used signatures in a database in order to prevent double spending.

Fair Blind Signatures

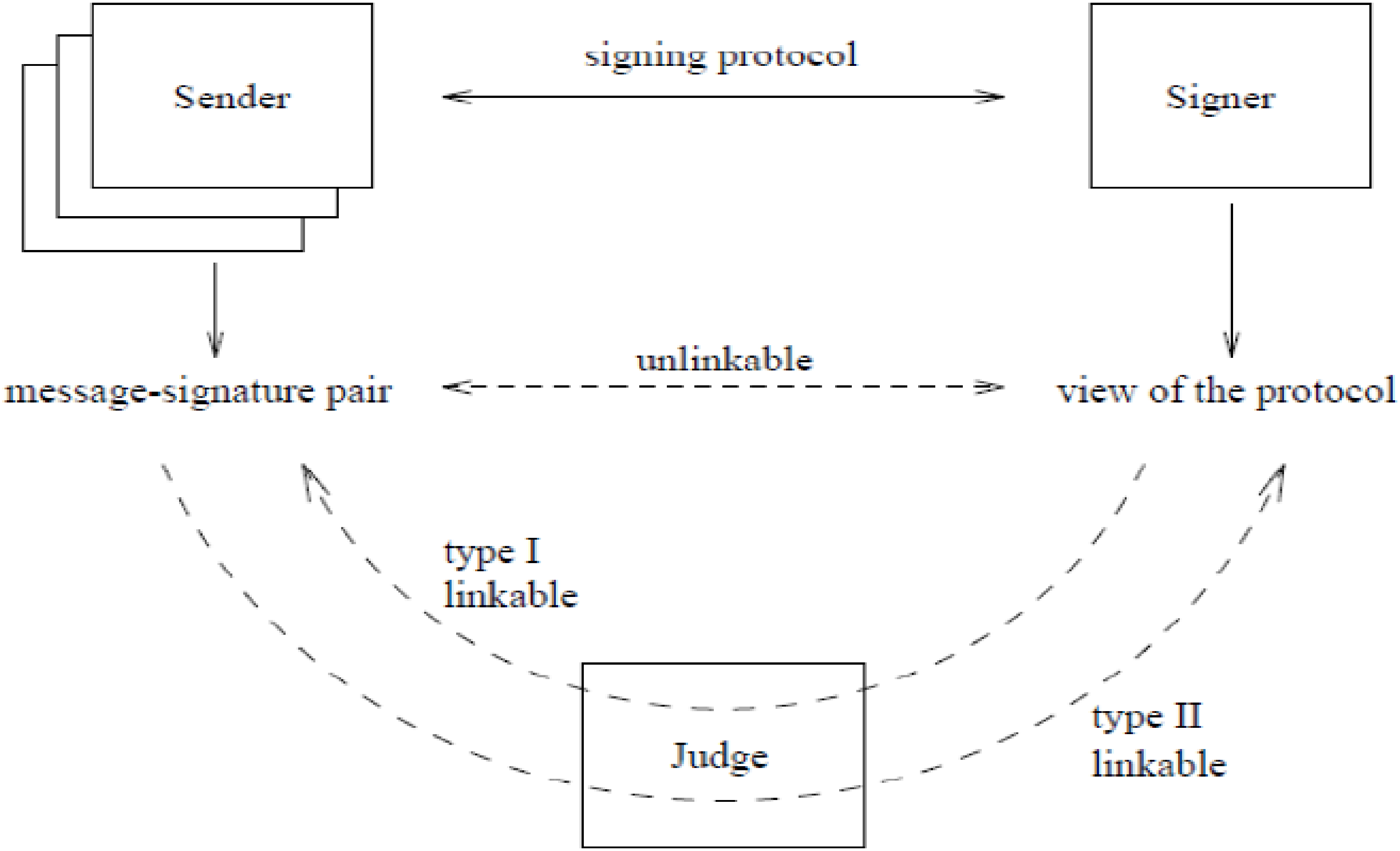
- A fair blind signature is a blind signature with revocable anonymity and unlinkability due to the involvement of a trusted third party, when this is required for legal reasons.

- Fair blind signature schemes have the addition property that a trusted third- party is involved, which possesses certain information that can help link a signer's view of the protocol to the message-signature pair, in case of a fraud or dishonest transaction such as blackmailing (as described in the example above) and money laundering.

The model can be visually summarized as follows:

Type 1:
Given the signer’s view of the protocol, **the judge delivers information that enables the signer** (or anyone else) to **efficiently recognize the corresponding message-signature pair** (i.e. the judge can extract the message that was signed).

Type 2:
Given the message- signature pair, **the judge delivers information that enables the signer to efficiently identify the sender of that message** or to find the corresponding view of the signing protocol



Identity- based Blind Signatures

- In traditional PKI-based digital signature schemes, certificates generated by a trusted third party are required to ‘bind’ the user’s identity and its public key.

- In identity-based cryptography, the only secret of each user is its secret identity (e.g. its IP or email address) as a secret key generated by a Key Generation Center (KGC). Hence, in such cryptosystems, the certificates and the intricate management can be avoided.

- Due to these advantages, ID-based blind signatures have been an active area of research. Research continues to be done in this arena to explore the advantages and short-comings of ID-based schemes to blind signatures

Application of Blind Signatures - voting

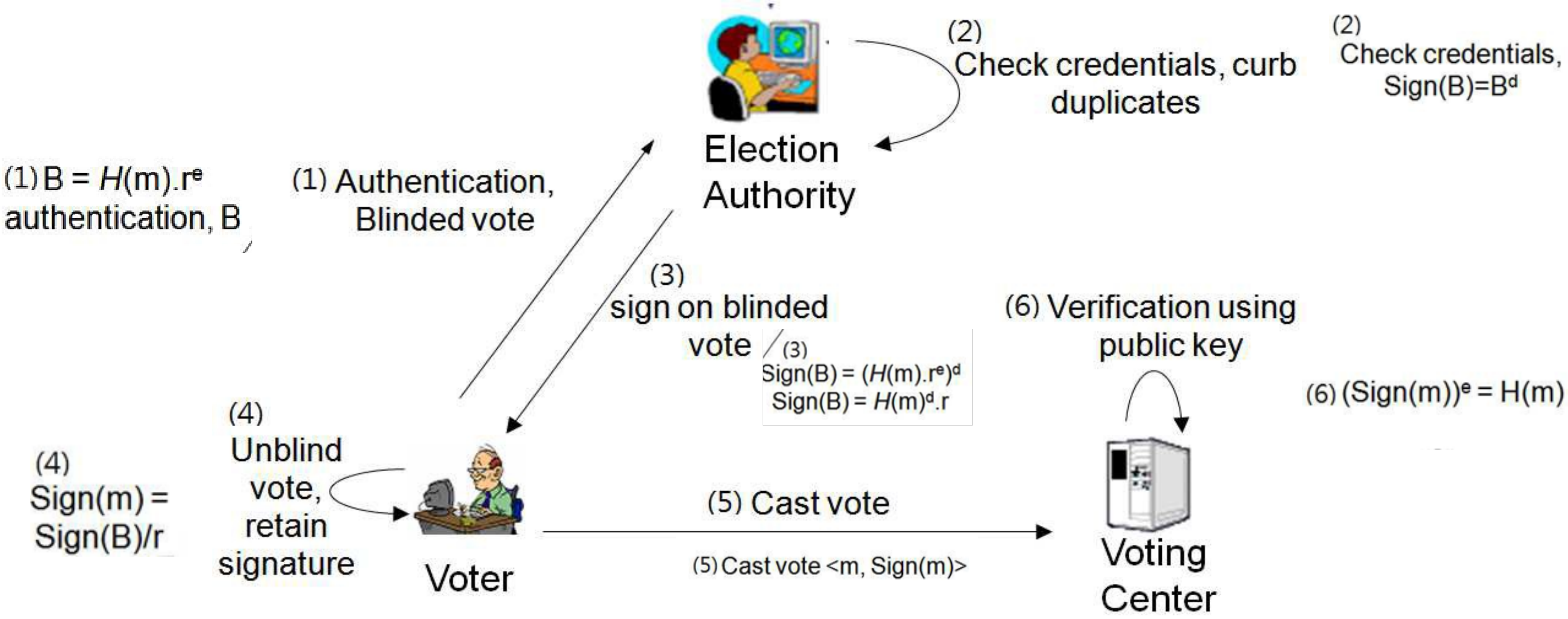
- A voter who presents identification is actually authenticating two things: his or her identity, and the validity of his or her vote.

- The integrity of some electronic voting system may require that each ballot be certified by an election authority before it can be accepted for counting.

- This allows the authority to check the credentials of the voter to ensure that they are allowed to vote, and that they are not submitting more than one ballot.

- Simultaneously, it is important that this authority not learn the voter's selections.

Application of Blind Signatures - voting



Election Authority

(1) $B = H(m).r^e$
authentication, B

(1) Authentication,
Blinded vote

(2)
Check credentials, curb duplicates

(2)
Check credentials,
$\mathrm{Sign}(B)=B^d$

(3)
sign on blinded vote

(3)
$\mathrm{Sign}(B) = (H(m).r^e)^d$
$\mathrm{Sign}(B) = H(m)^d.r$

(4)
Unblind vote, retain signature

(4)
$\mathrm{Sign}(m) = \mathrm{Sign}(B)/r$

Voter

(5) Cast vote

(5) Cast vote <m, Sign(m)>

Voting Center

(6) Verification using public key

(6) $(\mathrm{Sign}(m))^e = H(m)$

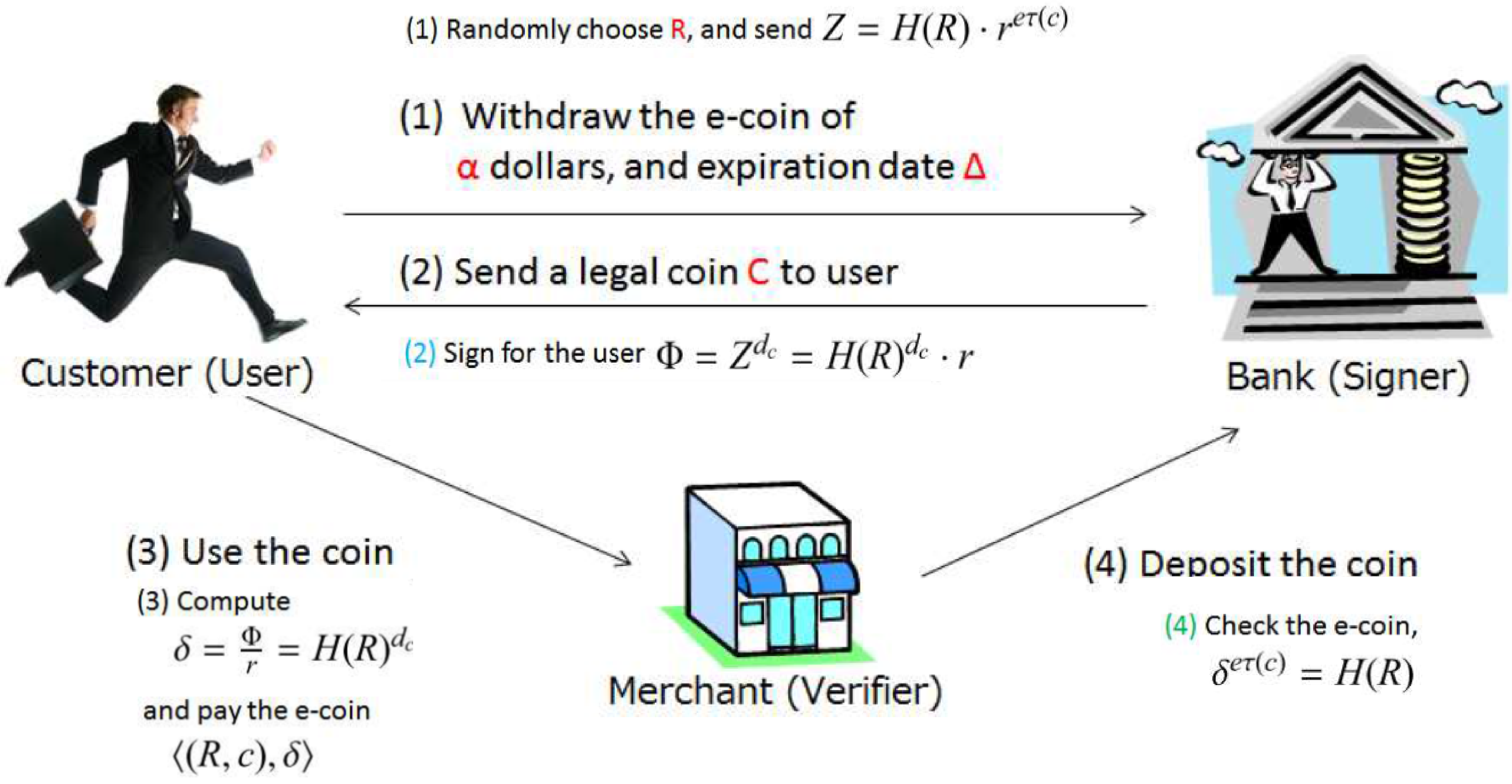
Application of Blind Signatures - Digital Cash Scheme

- The same sort of idea can be used to construct a digital analogue of cash.

- The key property of cash is anonymity: when you take money out of the bank, the bank gives you the cash without knowing what you buy, and when you spend money, the merchant has no idea who you are.

- By contrast, when you buy something with a credit card online, you have to tell the merchant who you are, and you have to tell the credit card company who you are making a purchase from. The potential for invasion of privacy is immense

Application of Blind Signatures - Digital Cash Scheme



(1) Randomly choose R, and send $Z = H(R) \cdot r^{e\tau(c)}$

(1) Withdraw the e-coin of α dollars, and expiration date Δ

(2) Send a legal coin C to user

(2) Sign for the user $\Phi = Z^{d_c} = H(R)^{d_c} \cdot r$

Customer (User)

Bank (Signer)

(3) Use the coin

(3) Compute

$$\delta = \frac{\Phi}{r} = H(R)^{d_c}$$

and pay the e-coin

$$\langle (R, c), \delta \rangle$$

Merchant (Verifier)

(4) Deposit the coin

(4) Check the e-coin,

$$\delta^{e\tau(c)} = H(R)$$