CS915/435 Advanced Computer Security - Elementary Cryptography

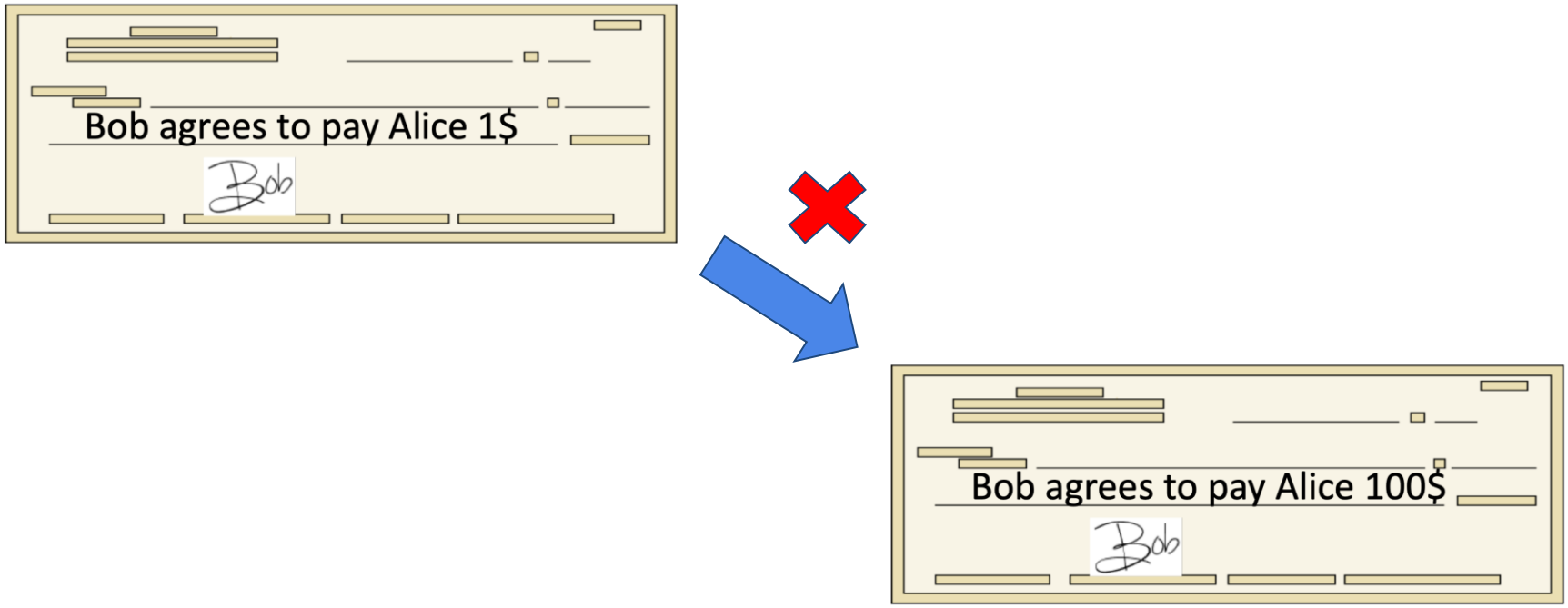
Digital signature

Roadmap

- Symmetric cryptography
  - Classical cryptographic
  - Stream cipher
  - Block cipher I, II
  - Hash
  - MAC
- Asymmetric cryptography
  - Key agreement
  - Public key encryption
  - **Digital signature**

Physical signatures

Goal: bind document to author

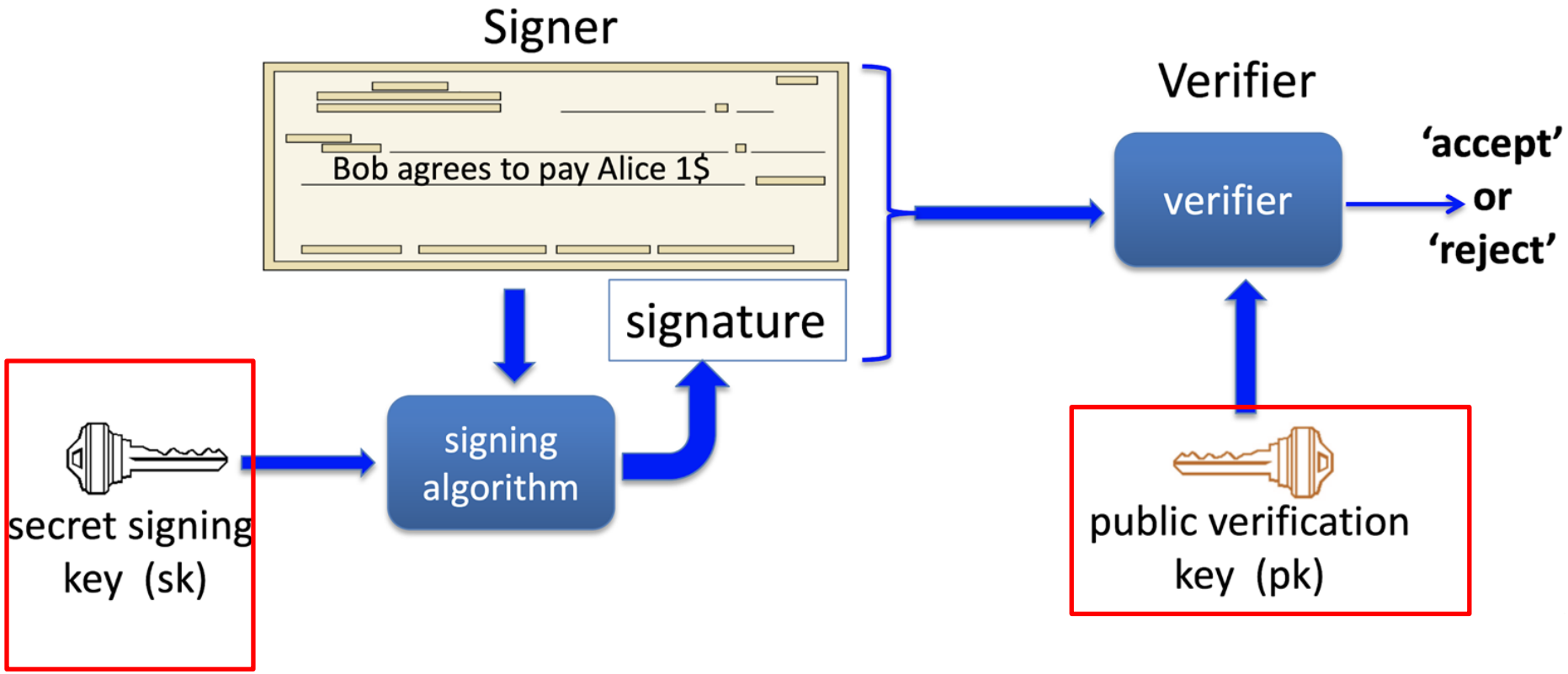


Problem in the digital world:

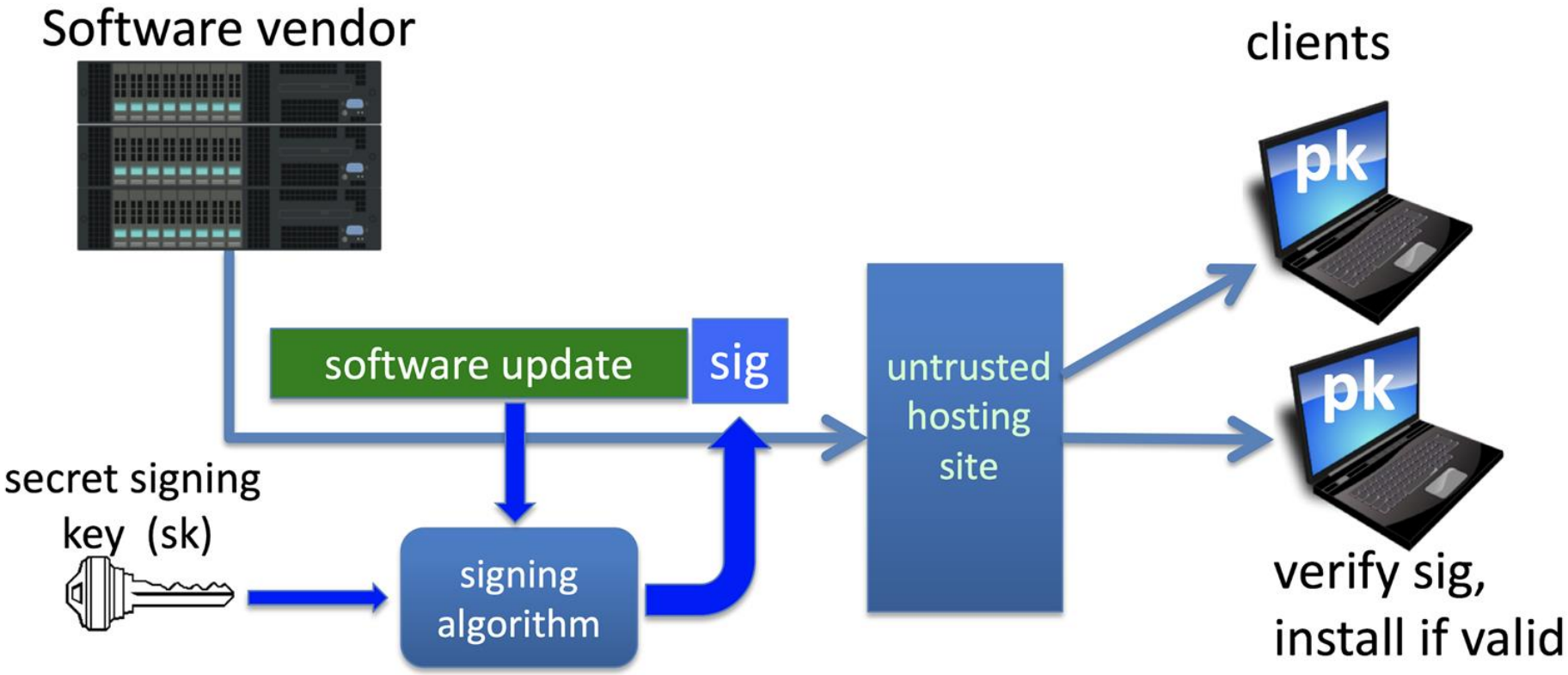
Anyone can copy Bob's signature from one doc to another

Digital signatures

Solution: bind signature with the document



A more realistic example



Another example: Update Distribution

Digital signatures: syntax

Def: a signature scheme (Gen, S, V)

- Gen(): randomized alg. outputs a key pair (pk, sk)
- S(sk, m) outputs signature σ
- V(pk, m, σ) outputs 'accept' or 'reject'

Consistency: for all (pk, sk) output by Gen:

$\forall\, m \in M$: V(pk, m, S(sk, m)) = 'accept'

Digital signatures: security

Attacker's power: **chosen message attack**

- For $m_1, m_2, \ldots, m_q$, attacker is given $\sigma \leftarrow S(sk, m_i)$

Attacker's goal: **existential forgery**

- Produce some **<u>new</u>** valid msg/sign pair (m, σ)

$$m \notin \{ m_1, \ldots, m_q \}$$

For a secure digital signature, attacker must not be able to produce a valid sig for a <u>new</u> message

Textbook RSA Signatures

KeyGen: $pk=(N, e)$, $sk=(N, d) \leftarrow$ **GenRSA**(1^n)

Sign: Given $sk=(N, d)$ and message m:

$$\sigma = m^d \quad (\textbf{mod } N)$$

Verify: Given $pk=(e, N)$ and signature σ:

$$m = \sigma^e \quad (\textbf{mod } N)$$

Is Textbook RSA Signature Secure?

- Security definition
  - Existential forgery

No-message Attack

Adversary **A** only has access to pk=(N, e).

How can he mount an attack?

A chooses some element σ in $\mathbf{Z}_N^*$.

A computes:

$$m = \sigma^e \ (\textbf{mod}\ N)$$

Output of **A**: (m, σ)

Selected-Message Attack

Adversary **A** has access to pk=(N, e) and can obtain two signatures from the signer.
How can **A** forge a signature on **any** chosen message m?

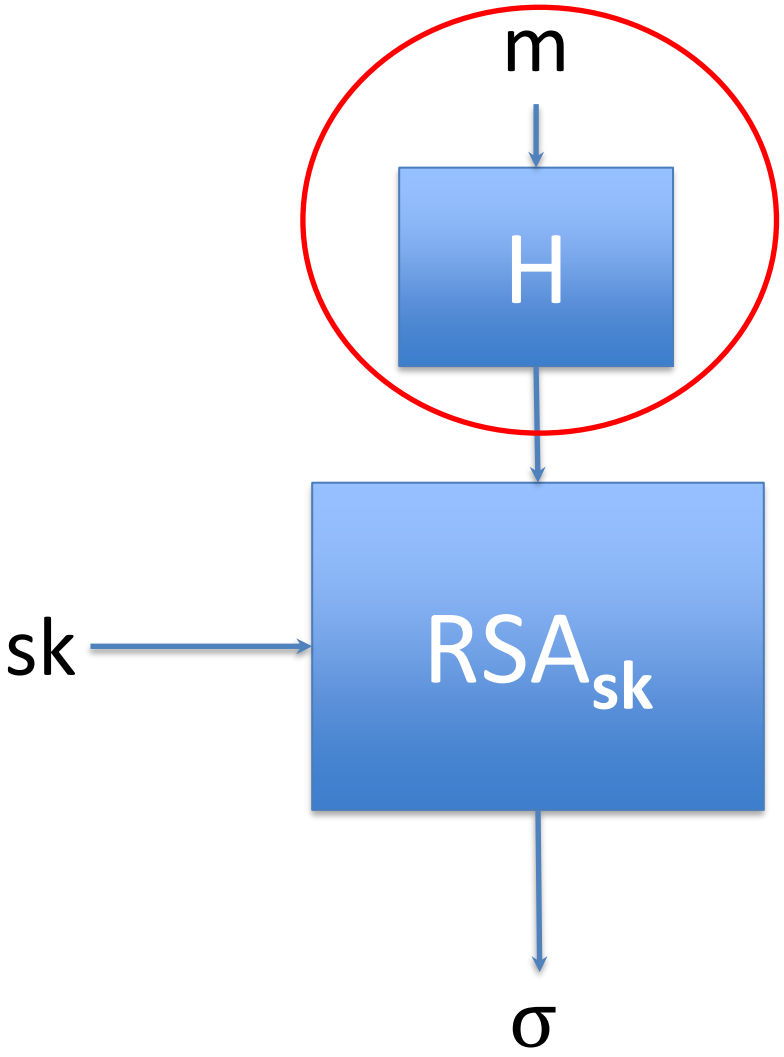
1. **A** chooses target message m.
2. **A** chooses a random message m_1.
3. Sets $m_2 = m/m_1$ (**mod** N)
4. **A** requests signatures σ_1 and σ_2 on m_1 and m_2.

$$\sigma_1 = m_1^d \qquad \sigma_2 = m_2^d$$

1. Result: $\sigma = \sigma_1 \cdot \sigma_2$ (**mod** N) is a valid signature on m.

$$\sigma_1 \sigma_2 = m_1^d m_2^d = (m_1 m_2)^d = m$$

Hashed RSA



(combined with secure encoding, e.g., RSA-PSS)

Case study

- In a company, a developer generates $N = p \times q$
- He then generates a unique signing key pair (sk, pk) for each employee in the company and distributes the keys through the company internal work.
  - $Sk = \{N, d\}$
  - $Pk = \{N, e\}$
- What could be the attacks?

Digital Signature Algorithm

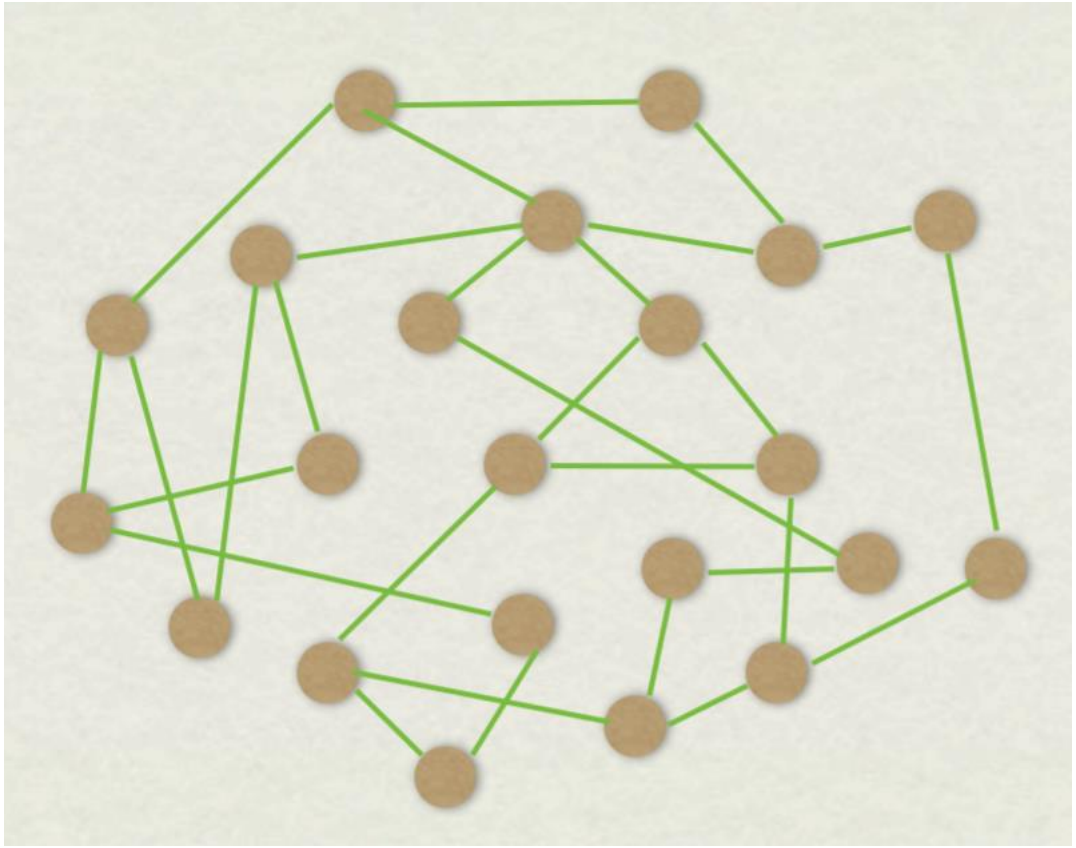
- Based on discrete logarithm instead
- Adopted as Digital Signature Standard in 1991
  - However, a controversial result
  - Schnorr signature was widely considered better
- Schnorr signature (1989)
  - Simpler than DSA
  - Has well-understood security proofs
  - DSA has no proofs

Schnorr signature

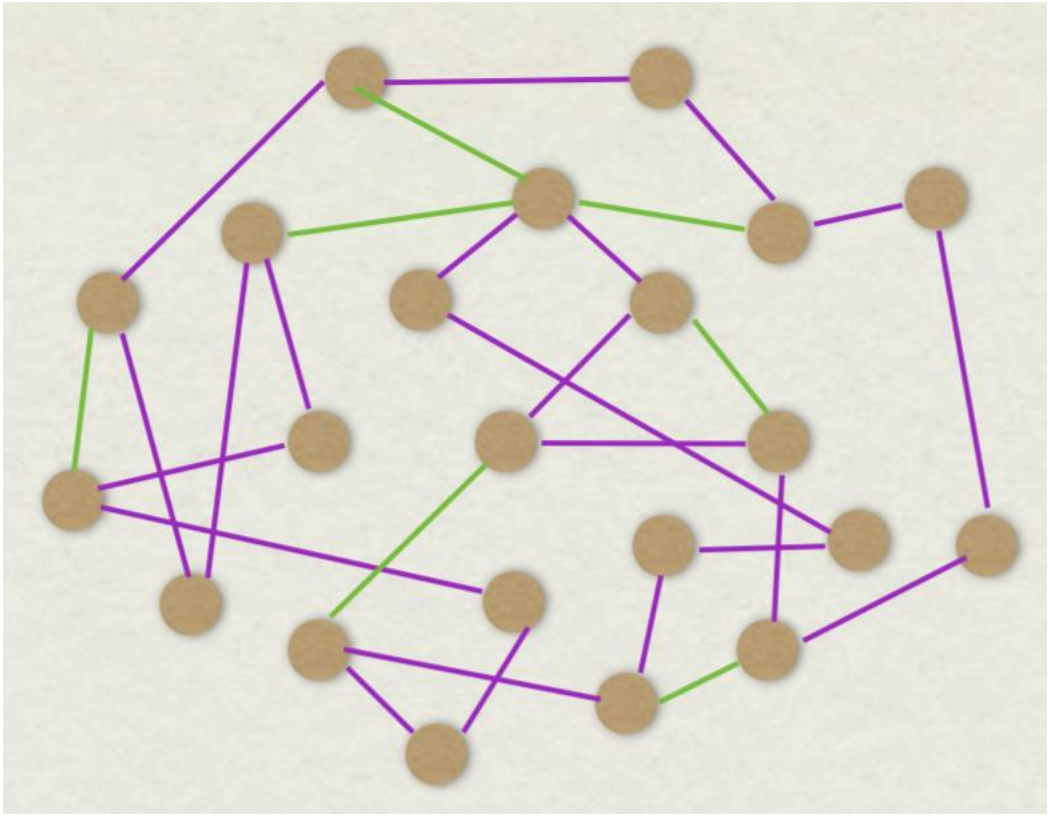
- A simple and elegant algorithm proposed by Claus Schnorr in 1989
- Its design is closely related to the notion of Zero Knowledge Proof

What is a zero-knowledge proof

- Imagine if I ask you to find a cycle in this graph



A simple proof by construction



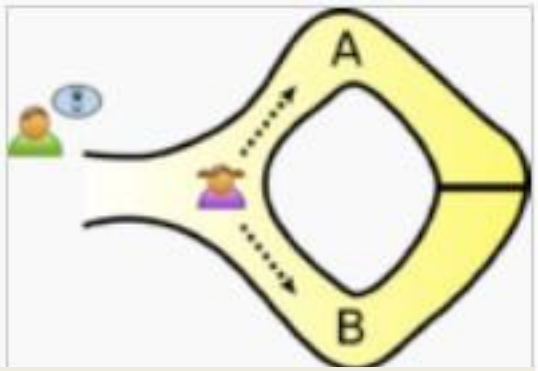
- Now you all know the answer

What is a zero-knowledge proof

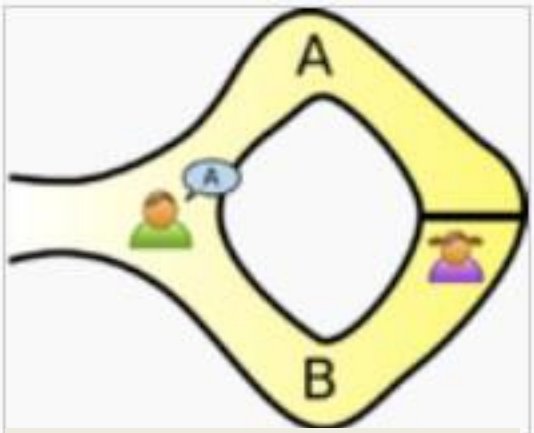
- A zero-knowledge proof proves that I know a secret without revealing the secret

An example of ZKP (interactive)

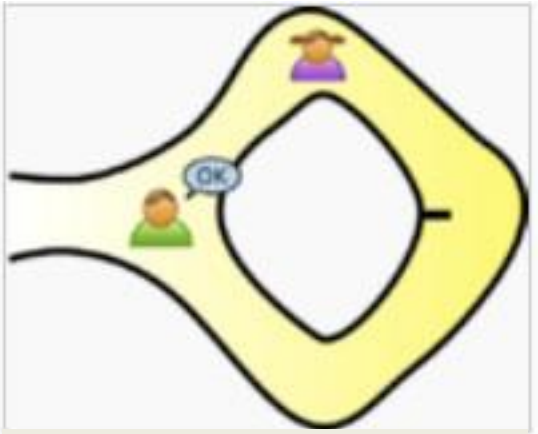
- Peggy wants to prove she has the secret key



Alice takes path A or B, while Bob waits outside



Bob chooses one side at random



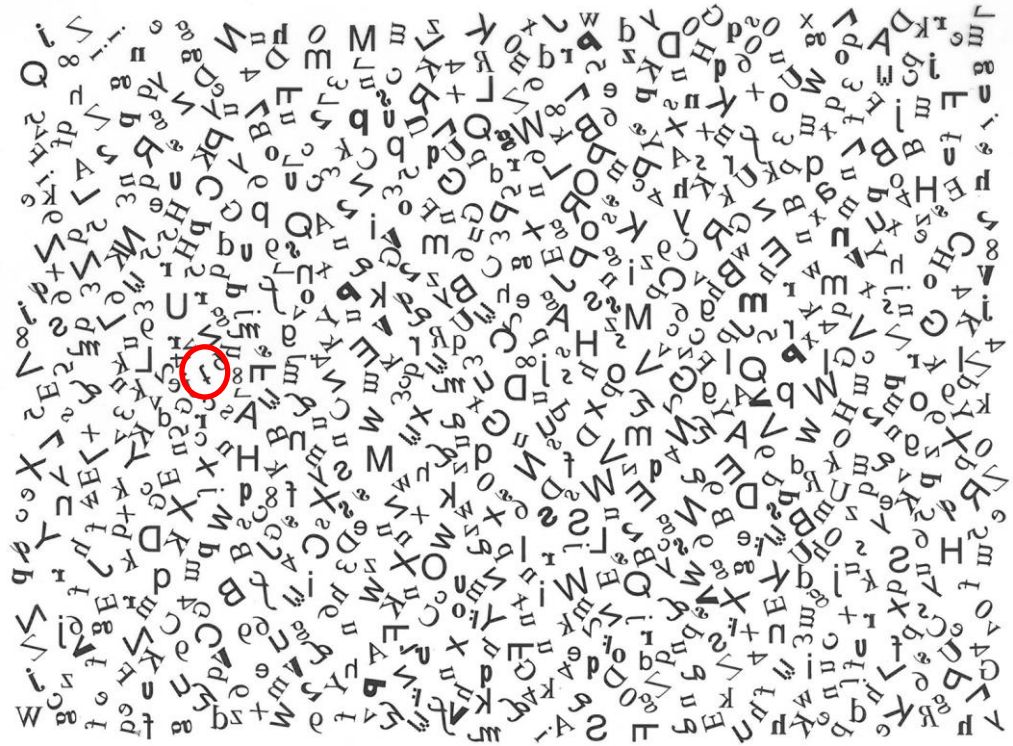
Alice appears from the correct side

Zero Knowledge Proof

- The verifier knows there is a 50% chance that the prover may cheat
- But if they repeat the protocol, the probability becomes 0.5^2 =25%
- If they repeat it 10 times, the probability is <0.1%
- And so on

Another example (non-interactive)

- Peggy wants to prove she knows the secret location of hidden treasure, marked by “t”



Requirements for ZKP

- Completeness
  - If the statement is true, the prover will be able to convince the verifier this fact
- Soundness
  - If the statement is false, no cheating prover can convince the verifier that it is true except with a small probability
- Zero-knowledge
  - A verifier learns nothing other than the fact that the statement is true

Schnorr identification Scheme

- Group Parameters
  - Let p and q be two large primes
  - q | p-1
  - Gq is the subgroup of Zp* of prime order q
  - g is a generator for the subgroup

$g^x \bmod p \longrightarrow X$

- User identification
  - Alice holds the private key x to a given public key X = g^x mod p where x is from [0, q-1]
  - Alice wishes to prove to Bob that she knows x

Interactive Zero Knowledge Proof

Alice		Bob
Choose random v from [0, q-1]		
Compute $V = g^v \bmod p$	V →	
	← c	Choose random c from $[0, 2^t-1]$
Compute $b = v - x*c \bmod q$	b →	Check if $V = g^b * (g^x)^c \bmod p$

$g^b * (g^x)^c \bmod p$

$g^{(v-x*c)} * (g^x)^c \bmod p$

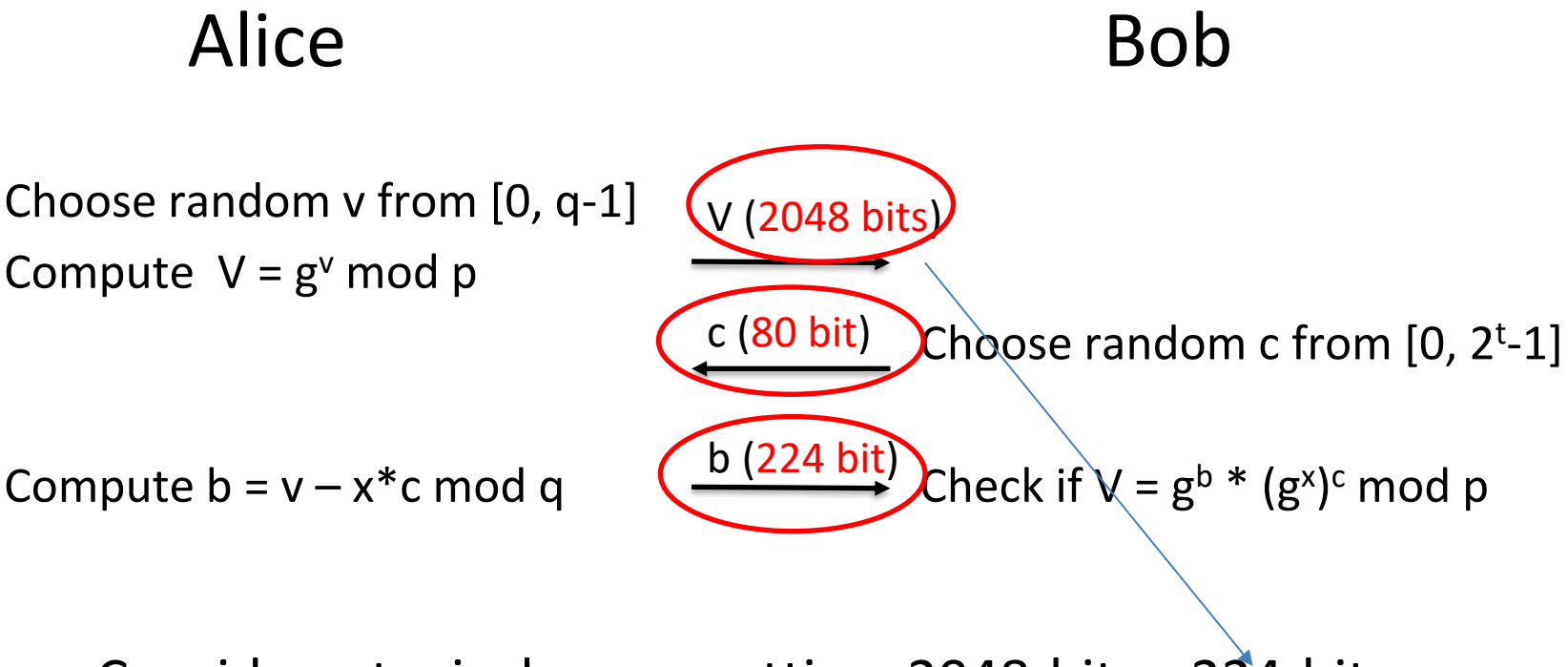
$g^v \bmod p$

Computational efficiency

- Designed to be very fast and efficient
- Alice Computation
  - One exponentiation (flow 1) - this can be precomputed offline before the scheme is executed
- Bob computation
  - One exponentiation (flow 3) - using a simultaneous exponentiation technique
- This is as efficient as you may possibly achieve for a public key scheme

$V = g^b * (g^x)^c \bmod p$

Communication efficiency



- Consider a typical group setting: 2048-bit p, 224-bit q
- We can further reduce the bits by using V' = SHA224(V) in the first flow. Then Bob checks if V' = SHA224($g^b * (g^x)^c$) mod p
- Hence, the communication efficiency is really close to the best you can hope for.

Completeness

If Alice knows the private key, she can pass the identification protocol successfully

Soundness

If Alice doesn't know the private key x, then the probability that she can pass the identification protocol successfully is 2^{-t} (t is the bit length of c)

Suppose Alice guessed the correct value of the challenge c. She can pass the identification by choosing an arbitrary value b, and pre-compute the following value for flow 1

$$V = g^b * (g^x)^c$$

Proof of soundness (sketch)

$[0, 2^t - 1]$

Assume Alice can guess more than one c values to pass the identification. I.e., she knows c_1, c_2, b_1 and b_2 such that

$$V = g^{b1} (g^x)^{c1} = g^{b2} (g^x)^{c2}$$

It follows that

$$g^{b1-b2} = (g^x)^{c2-c1}$$

$$b_1 - b_2 = x(c_2 - c_1)$$

We can conclude that Alice knows x, since

$$x=(b_1-b_2)/(c_2-c_1)$$

Zero-knowledge

Alice proves her knowledge of the private key x, without revealing any information about x.

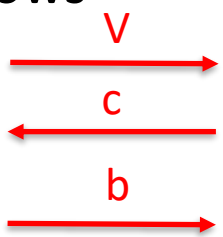
Proof of zero-knowledge (sketch)

Assume an **honest verifier**, who chooses the challenge c at random.

A **transcript** of a session consists of (V, c, b). For any fixed challenge c, there is a one-to-one correspondence between V and b.

Alice (or anyone else) can generate **simulated** transcripts as follows

- Choose c uniformly at random from $\{0,\ldots,2^t-1\}$
- Choose b uniformly at random from $\{0,\ldots,q-1\}$
- Compute $V = g^b * (g^x)^c \bmod p$



$V = g^b * (g^x)^c \bmod p$

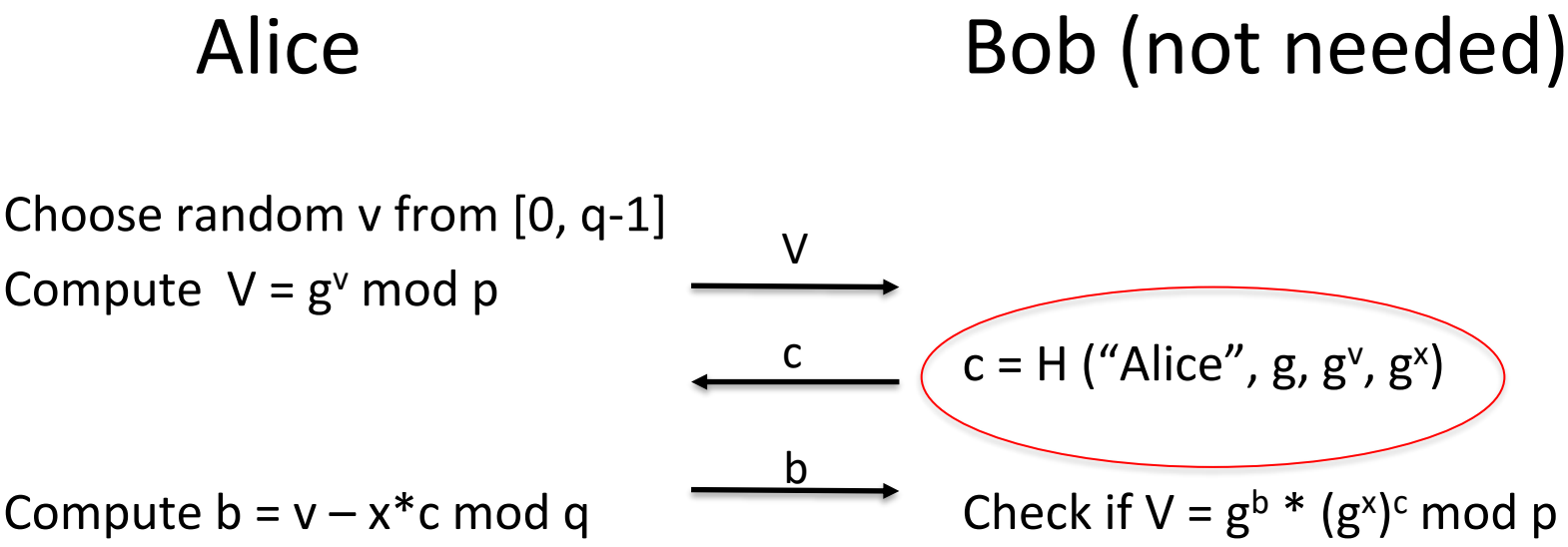
The simulated transcript is indistinguishable from a real transcript

Chapter 9.4, "Cryptography: Theory and Practice", Stinson. 3rd, 2006
http://www.ksom.res.in/files/RCCT-2014-III-CM/CryptographyTheoryandpractice(3ed).pdf

Non-interactive ZKP

- Replace the verifier with the results of a cryptographic hash function
- Naturally fulfills the honest-verifier requirement
- A technique called Fiat-Shamir heuristic

Schnorr Non-interactive Proof



Notes

1. **Bob is no longer needed since Alice can issue the challenge c by herself**
2. **The prover's identify often needs to be included in the hash**

Schnorr signature

Derived from Schnorr non-interactive proof

Key pair: private key x, public key $X = g^x \bmod p$

Signing a message m

1. Choose a random v, compute $V = g^v \bmod p$
2. Let $h=H(V \,||\, m)$, $s = v-xh$
3. Signature is the pair (V, s)

Verification

- Check $g^s X^h == V$?

Importance of ZKP

- Schnorr signature is one example of ZKP
- Widely used in cryptographic protocols to ensure participants follow the specification honestly

"Do not assume that a message you receive has a particular form (such as g^r for known r) unless you can check this."

- The sixth principle by Ross Anderson and Roger Needham

Anderson and Needham, "Robustness principles for public key protocols", Crypto'95

DSS Key Generation

GenDSS(1^n)

Input: key length *n*

Create a cyclic group G, sub-group of $(\mathbf{Z}_p)^*$ with generator g with prime-order q.

q divides p-1.

Choose random x in $\mathbf{Z}_q$

Compute $y = g^x$ (**mod** q)

Output: *pk*=(p, q, g, y), sk=(p, q, g, x)

Digital Signature Standard

KeyGen: $pk=(p, q, g, y)$, $sk=(p, q, g, x) \leftarrow$ **GenDSS**(1^n)

Sign: Given $sk=(p, q, g, x)$ and message m:

Choose k random in $\mathbf{Z}_q$; $r = g^k$ (**mod** p)(**mod** q)

$\sigma = (r, s=(\mathbf{H}(m) + xr) \cdot k^{-1}$ (**mod** q))

Verify: Given $pk=(p, q, g, y)$, m and signature σ:

Compute $u_1 = H(m) \cdot s^{-1}$ (**mod** q) and

$u_2 = r \cdot s^{-1}$ (**mod** q)

Check $r = g^{u1}\, y^{u2}$ (**mod** p) (**mod** q)

Correctness of DSS

We call **m** := **H**(m). **We have** $y = g^x$

$r = g^k \pmod p \pmod q$

$s = (\mathbf{m} + xr) \cdot k^{-1} \pmod q$

$g^{ms-1}\, y^{rs-1} = g^{ms-1}\, (g^x)^{rs-1} = g^{(m\,s-1) + (xr\,s-1)}$ **| all (mod** p)

"Working in the exponent" $\mathbf{Z}_q$**:| all (mod** q)

$(\mathbf{m}\, s^{-1}) + (xr\, s^{-1}) = (\mathbf{m} + xr)\, s^{-1}$

$= (\mathbf{m} + xr)\, ((\mathbf{m} + xr) \cdot k^{-1})^{-1} = k$

Summary DSS

- The Digital Signature Standard is an **international standard** (proposed by NIST)
- Derived from Schnorr signature (patented then, but expired now)
- Widely used in practice.
- It has been scrutinized for years w/o any attack being found.
- **No security proof** exists.