Cryptography

Lecture 12

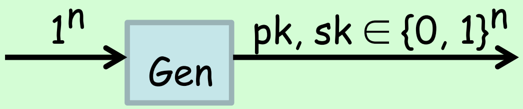
Arpita Patra

Digital Signatures

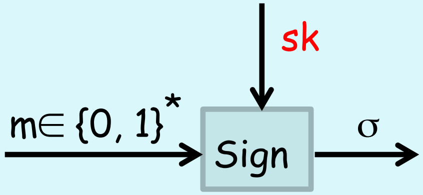
- In PK setting, privacy is provided by PKE

- Integrity/authenticity is provided by digital signatures (counterpart of MACs in PK world)

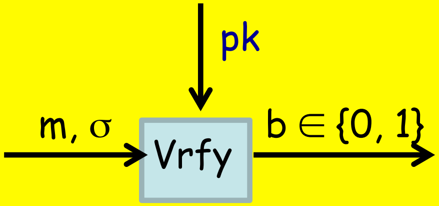
- **Definition:** A Digital signature scheme Π consists of three PPT algorithms (Gen, Sign, Vrfy):



- Randomized
- pk: public key (verification key)
- sk: private key (signing key)



- Usually Randomized
- σ is signature for m



- Deterministic
- b = 0 → invalid signature
- b = 1 → valid signature

- (pk, sk) plays a different "role" compared to public-key encryption

  >> sk – signature generation (whereas pk was used for ciphertext generation)

  >> pk – public verification of the signature (whereas sk was used for decryption)

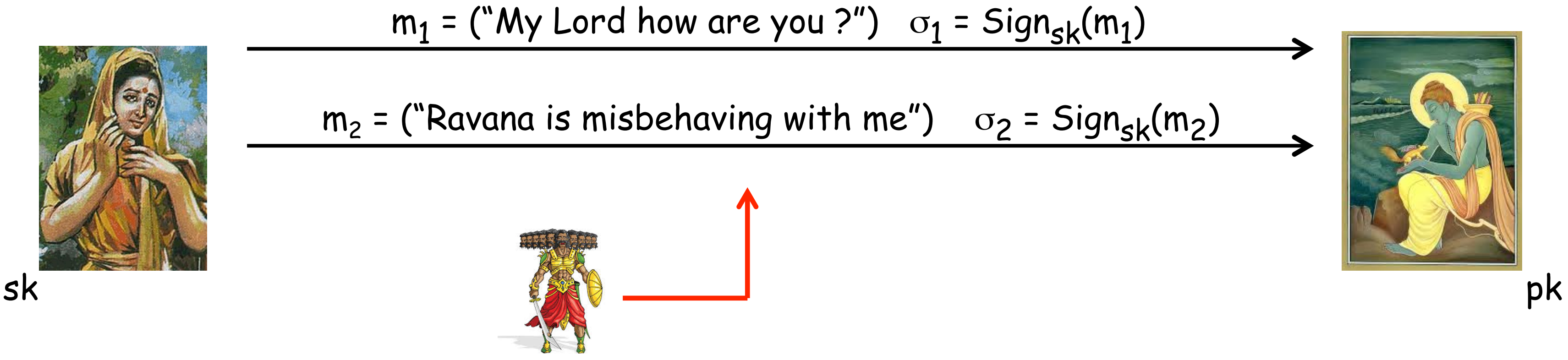
- Signatures cannot be obtained by "reversing" a public-key encryption scheme

- Correct ness: Except with a negligible probability over (pk, sk) output by Gen(1^n), we require the following for every (legal) plaintext m

$$\mathrm{Vrfy}_{pk}(m, \mathrm{Sign}_{sk}(m)) = 1$$

Digital Signatures : Security

- ❑ Goal: we want to prevent a situation like the following: Π = (Gen, Sign, Vrfy)



m_1 = ("My Lord how are you ?") $\sigma_1 = \mathrm{Sign}_{sk}(m_1)$

m_2 = ("Ravana is misbehaving with me") $\sigma_2 = \mathrm{Sign}_{sk}(m_2)$

sk

pk

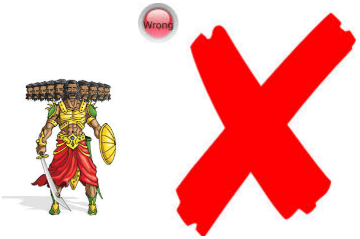
Digital Signatures : Security

❑ Goal: we want to prevent a situation like the following: Π = (Gen, Sign, Vrfy)

sk

m'_1 = ("Ravana is not that bad") $\sigma'_1 = \mathrm{Sign}_{sk}(m'_1)$

m'_2 = ("I am fine here") $\sigma'_2 = \mathrm{Sign}_{sk}(m'_2)$



pk

❑ How to model the above requirement via security experiment ? --- Experiment Sig-forge$_{A,\Pi}$(n)

PPT Attacker A

I can forge Π

pk

$m_1, \ldots, m_q$

$\sigma_1, \ldots, \sigma_q$ $\sigma_i \leftarrow \mathrm{Sign}_{sk}(m_i)$

(m^*, σ^*)

Let me verify

pk, sk

Gen(1^n)

Π is existentially-unforgeable/CMA if for every PPT A:

$$\Pr\left[\text{Sig-forge}_{A,\Pi}(n) = 1\right] \leq \mathrm{negl}(n)$$

$$b = \begin{cases} 1 \text{ if } \mathrm{Vrfy}_{pk}(m^*, \sigma^*) \neq 0 \text{ and } (m^*, \sigma^*) \notin \{(m_i, \sigma_i)\} \\ 0 \text{ otherwise} \end{cases}$$

MAC vs Digital Signature

MAC	Digital Signature
- Key distribution has to be done apriori.	Not completely correct! Relies on the fact that there is a way to send the public key in an authenticated way to the verifiers
- In multi-verifier scenario, a signer/prover need to hold one secret key for every verifier	+ One signer can setup a single public-key/secret key and all the verifiers can use the same public key
- Well-suited for closed organization (university, private company, military). Does not work for open environment (Internet Merchant)	+ Better suited for open environment (Internet) where two parties have not met personally but still want to communicate securely (Internet merchant & Customer)
+ Very fast computation. Efficient Communication. Only way to do auth in resource-constrained devices such as mobile, RFID, ATM cards etc	- Orders of magnitude slower than Private-key. Heavy even for desktop computers while handling many operations at the same time
- NO Public Verifiability & Transferability	+ Public Verifiability & Transferability
- NO Non-repudiation (cannot deny only to the person holding the key)	+ Non-repudiation (cannot deny to anyone)

Some Results on Digital Signatures

- Feasibility Results for DS: Unlike PKE (which needs more assumption than HF/OWF), DS can be constructed just based on HF (in fact just from OWF) [Rompel STOC'90]

- DS Schemes in Practice:

  >> RSA-FDH (Full Domain Hash) - RSA Assumption + HF – PKCS #1 v2.1

  >> Digital Signature Algorithm (DSA)- DL + HF- Digital Signature Standard (DSS)

Digital Certificates and Public-key Infrastructure (PKI)

Public-key World

Is pk_S indeed a genuine public-key of Sita ?

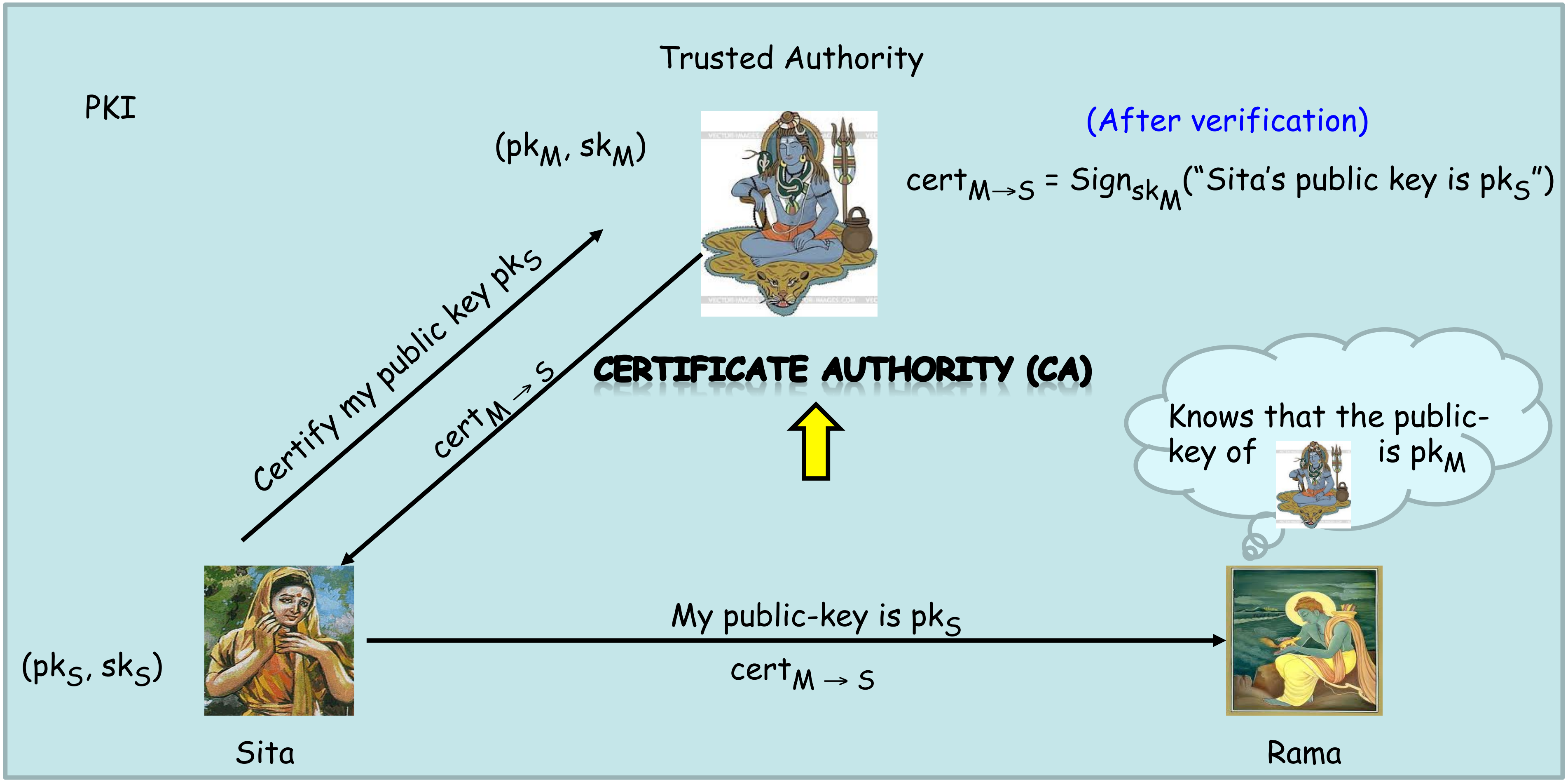
(pk_S, sk_S)

My public-key is pk_S

Sita

Rama

Digital Certificates and Public-key Infrastructure (PKI)



PKI

Trusted Authority

(pk_M, sk_M)

(After verification)

$cert_{M \to S}$ = $Sign_{sk_M}$("Sita's public key is pk_S")

Certify my public key pk_S

$cert_{M \to S}$

CERTIFICATE AUTHORITY (CA)

Knows that the public-key of is pk_M

My public-key is pk_S

$cert_{M \to S}$

(pk_S, sk_S)

Sita

Rama

pk_S is a genuine public key if and only if

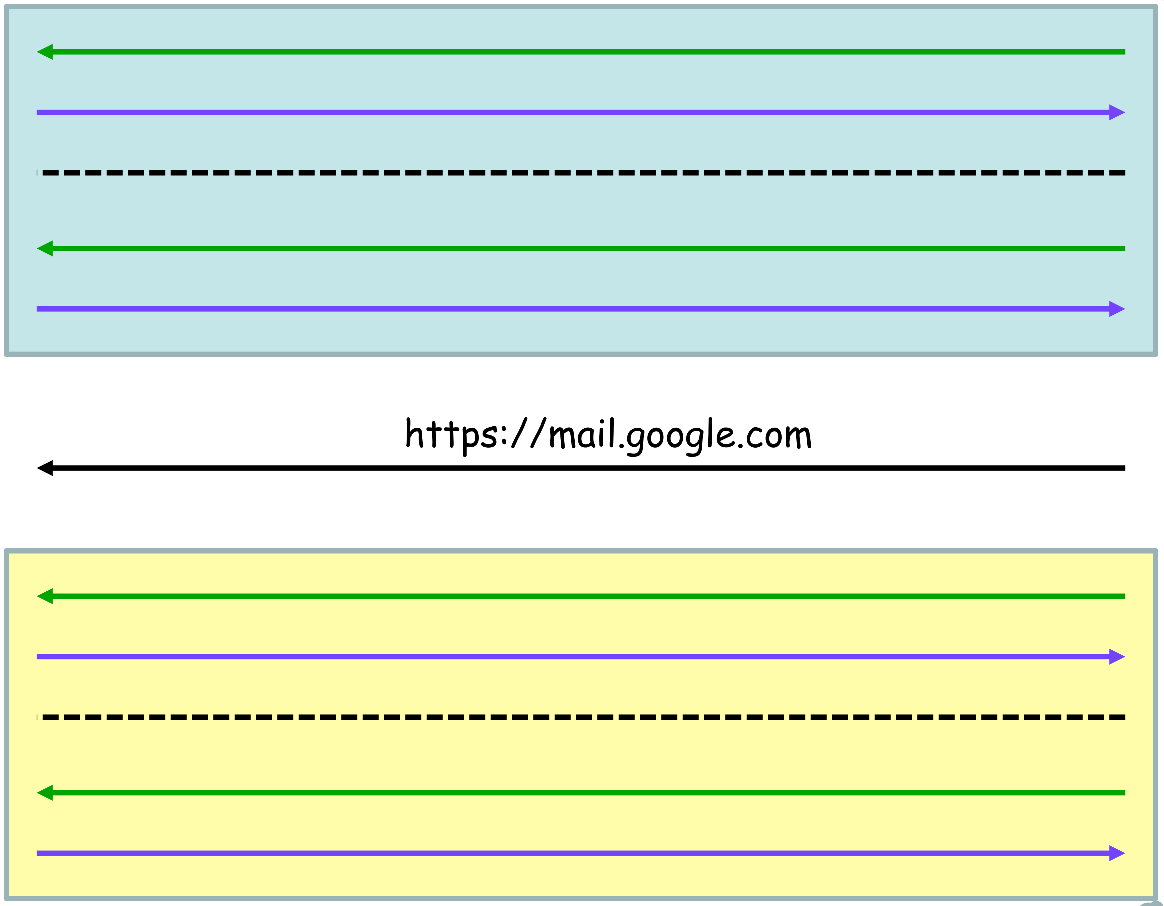
$Vrfy_{pk_M}$("Sita's public key is pk_S", $cert_{M \to S}$") = 1

- ❑ Several types of PKI used in practice
  - ➢ Single CA, multiple CA, PGP, etc
- ❑ Public keys of CA are pre-configured in web browsers
  - ➢ Programmed to verify the certificates issued by those CAs

Putting It All Together - TLS (Transport Layer Security)

Handshake protocol

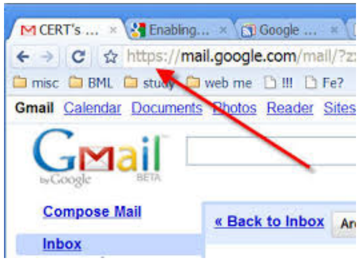
Authenticated Key Exchange



(Public-key crypto)

https://mail.google.com

Server



Client

(Private-key crypto)

Authenticated Private Communication
(Using keys established by handshake protocol)

Record-layer protocol

Putting It All Together - SSL/TLS
(The Handshake Protocol)

(pk_1, sk_1)

CA_1

(pk_2, sk_2)

CA_2

(pk_3, sk_3)

CA_3

(pk_4, sk_4)

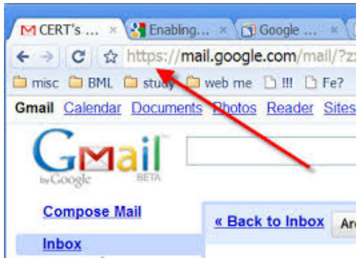
CA_4

$\text{cert}_{2 \to S}$

Certifying that pk_S is the public key of the server

Server
(pk_S, sk_S)

pk_1, pk_2, pk_3, pk_4
(pre-configured)



Client

Putting It All Together – SSL/TLS (The Handshake Protocol)

(pk_1, sk_1) (pk_2, sk_2) (pk_3, sk_3) (pk_4, sk_4)

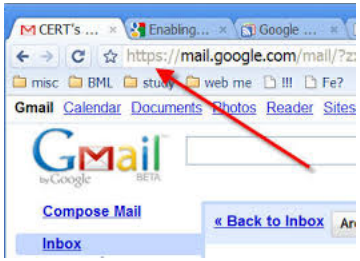
CA_1 CA_2 CA_3 CA_4

transcript'

$cert_{2\rightarrow S}$

(pk_S, sk_S)

pk_1, pk_2, pk_3, pk_4 (pre-configured)



CERT's ...
Enabling...
Google ...
https://mail.google.com/mail/?z
misc BML study web me !!! Fe?
Gmail Calendar Documents Photos Reader Sites
Gmail by Google BETA
Compose Mail
Inbox
« Back to Inbox

Supported ciphersuites (hash functions, block ciphers, etc), N_C

$cert_{2\rightarrow S}$

Corresponding ciphersuites, N_S, pk_S,

c

$\tau_C := Mac_{mk}(transcript)$

Random nonce N_C

Random nonce N_S

$\tau_S := Mac_{mk}(transcript')$

$Vrfy_{pk_2}(pk_S, cert_{2\rightarrow S}) \overset{?}{=} 1$

$pmk := Decaps_{sk_S}(c)$

$(c, pmk) \leftarrow Encaps_{pkS}(1^n)$

$mk := KDF(pmk, N_c, N_s)$

$mk := KDF(pmk, N_c, N_s)$

$k_C, k'_C, k_S, k'_S := PRG(mk)$

Agreed symmetric keys

$k_C, k'_C, k_S, k'_S := PRG(mk)$

$Vrfy_{mk}(transcript, \tau_C) \overset{?}{=} 1$

$Vrfy_{mk}(transcript', \tau_S) \overset{?}{=} 1$

Putting It All Together - SSL/TLS (The Record-layer Protocol)

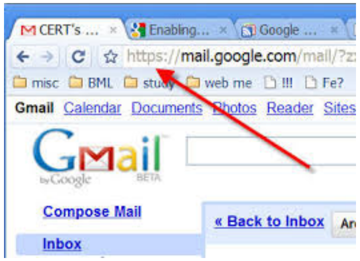
k_C, k'_C, k_S, k'_S

Authenticated communication

(k_S, k'_S)

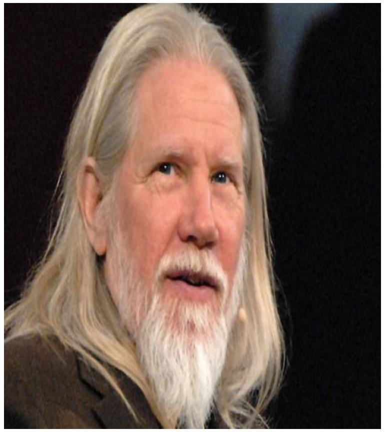
Authenticated communication

(k_C, k'_C)



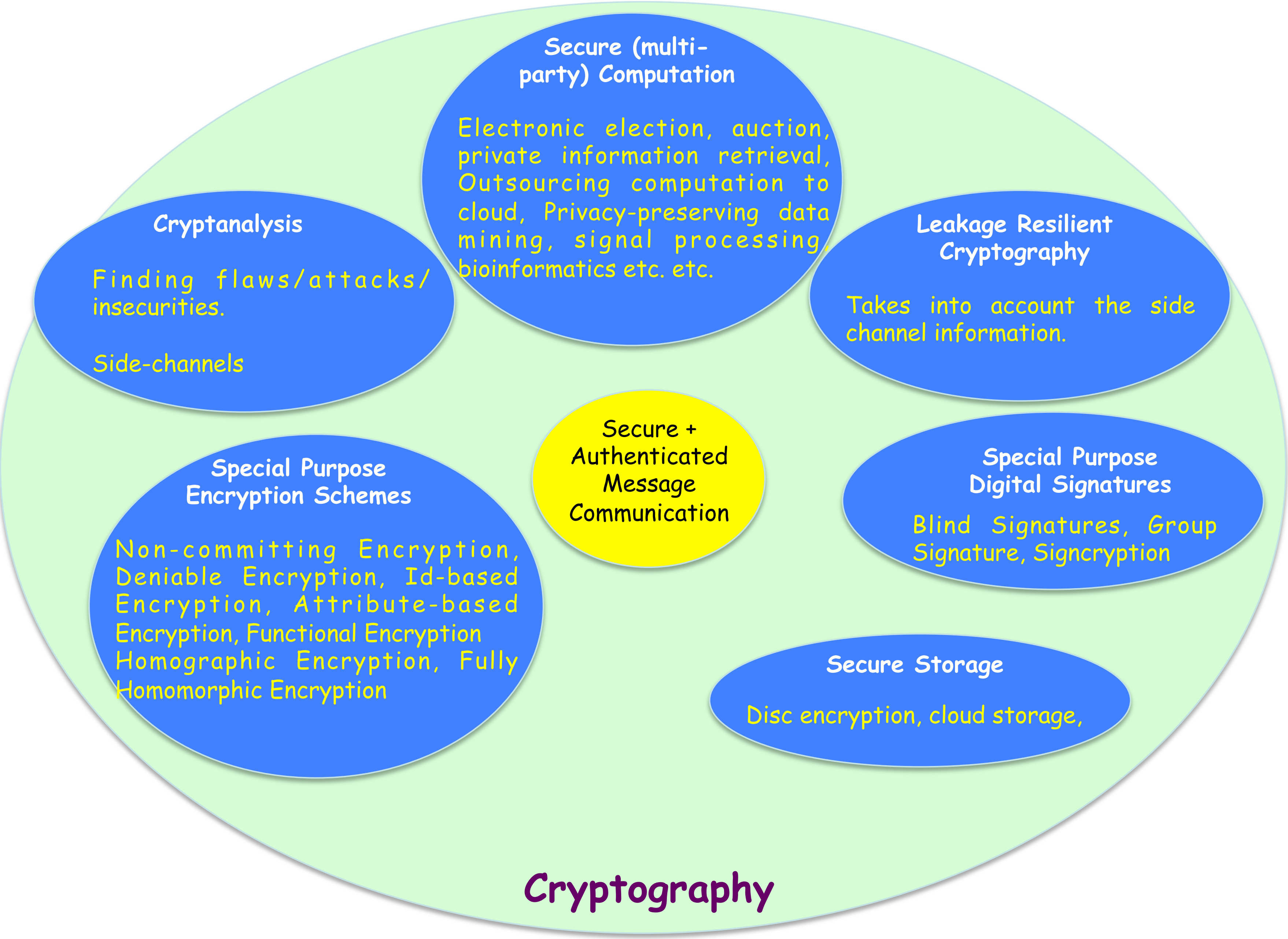
k_C, k'_C, k_S, k'_S

Public Key Cryptography



Whitfield Diffie, Martin E. Hellman:
New directions in cryptography.
IEEE Transactions on Information Theory 22(6): 644-654 (1976)

What We have seen and not seen?



Secure (multi-party) Computation

Electronic election, auction, private information retrieval, Outsourcing computation to cloud, Privacy-preserving data mining, signal processing, bioinformatics etc. etc.

Cryptanalysis

Finding flaws/attacks/ insecurities.

Side-channels

Leakage Resilient Cryptography

Takes into account the side channel information.

Secure + Authenticated Message Communication

Special Purpose Encryption Schemes

Non-committing Encryption, Deniable Encryption, Id-based Encryption, Attribute-based Encryption, Functional Encryption Homographic Encryption, Fully Homomorphic Encryption

Special Purpose Digital Signatures

Blind Signatures, Group Signature, Signcryption

Secure Storage

Disc encryption, cloud storage,

Cryptography

Crypto Zoo

We will get Cryptomaniac next semester with course on Secure Computation

Cryptomania: Everything that u can design in Crypto

Oblivious Transfer

Secret Sharing

Commitment Schemes

Zero Knowledge Proofs

Public Key Encryption

Hash Functions

SPRP

PRG

MAC

Minicrypt: SKC, Digital Signatures

One way permutation

One way Function

Choice is yours; whether u want to confine yourself in Minicrypt or u want turn to a Cryptomaniac.

Course on Secure Computation

Primitives	Definition Paradigms	Proof Paradigms
>> Oblivious Transfer	>> Real World- Ideal World Paradigm	>> Black-box Reduction
>> Commitment Schemes	>> Universal Composability (UC) Paradigm	>> Non-black-box reduction
>> Zero Knowledge Proofs		>> Random-Oracle Model (ROM)
>> Secret Sharing		
>> Threshold Encryption		
>> Secure Computation in various setting		
>> Secure Computation of Practical Problems- Set Intersection, Genomic Computation		
>> Byzantine Agreement & Broadcast		

- For many constructions based on HF
- Modeled as a random oracle (a truly random function from X → K)
- Access to H is via oracle calls
  - To compute H(a), call oracle with a, who returns a random value from co-domain as the output --- once a value is associated as H(a), the association remains fixed for future instances
- Calls to the oracle are private
  - If attacker has not queried for H(a), then H(a) remains uniformly random for the attacker

Concluding Remarks

Thank You!

El Gamal like KEM

Gen(1^n)

(G, o, q, g)

h = g^x. For random x

pk= (G,o,q,g,h,H), sk =

CPA-secure KEM +
COA-secure SKE =>
CPA-secure PKE @
COA-secure SKE

$Dec_{sk}(c)$

k = $H(c^x) = H(g^{xy})$

Security 1

CDH
(Weaker than DDH; hard to compute g^{xy} even given g^x, g^y)

+

H is "Random Oracle"
(Random => H behaves like an ideal random function)

Security 2

HDH- Hash Diffie-Hellman
(Weaker than DDH but stronger than CDH when Hash function is implemented using known practical ones; hard to distinguish $H(g^{xy})$ from a random string $\{0,1\}^m$ even given g^x, g^y) where H: $\{0,1\}^* \to \{0,1\}^m$

+

No assumption on H. It is incorporated in the above

Security 3

DDH
(Strongest Diffie-Hellman Assumption; hard to distinguish g^{xy} from a random group element even given g^x, g^y)

+

"Regular" H
(Regular => The number of elements from G that maps to k is approximately the same for all k)