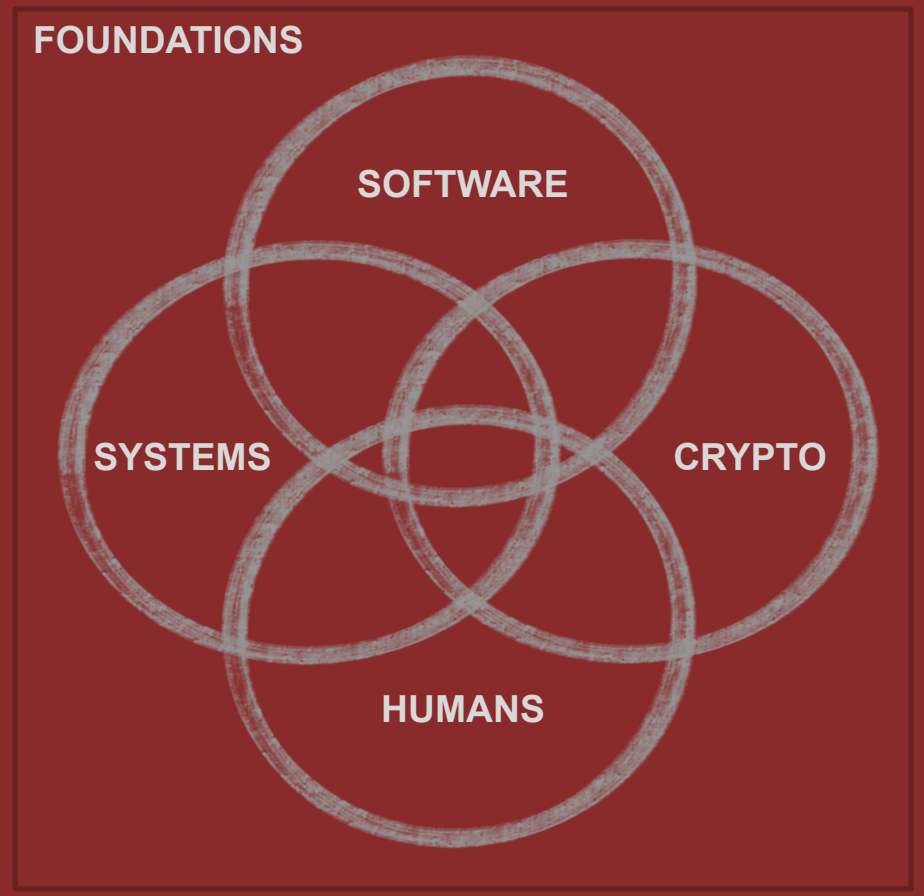
Διάλεξη #18 - Authenticated Encryption and Asymmetric Crypto

Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών

Εισαγωγή στην Ασφάλεια

Θανάσης Αυγερινός



Huge thank you to David Brumley from Carnegie Mellon University for the guidance and content input while developing this class (lots of slides from Dan Boneh @ Stanford!)

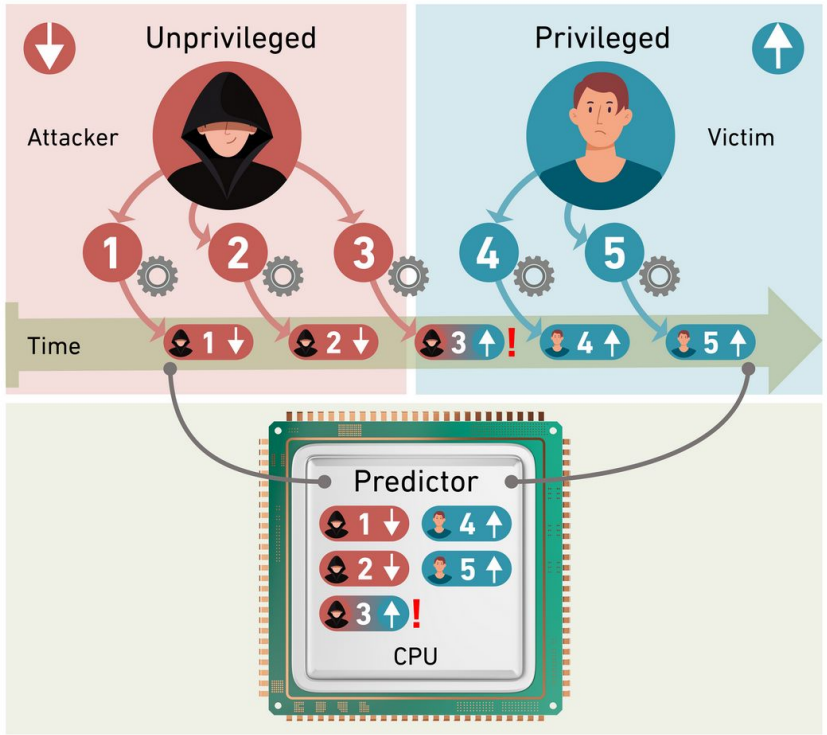
Την προηγούμενη φορά

- Hashes Intro
- Hash Constructions
- HMAC
- Hash Tricks/Datastructures

Security in the News

ETH Zurich researchers discover new security vulnerability in Intel processors

Computer scientists at ETH Zurich discover new class of vulnerabilities in Intel processors, allowing them to break down barriers between different users of a processor using carefully crafted instruction sequences. Entire processor memory can be read by employing quick, repeated attacks.



[Meltdown and Spectre] attack mitigations led to decreased performance and increased cost (by up to ~30%)

Critical BitLocker Flaw Exploited in Minutes: Bitpixie Vulnerability Proof of Concept Unveiled

Cyber Security *Cyber Security News* *Microsoft* *Vulnerability*

PUBLISHED ON **MAY 15, 2025** BY **KAAVIYA**

Unlike traditional hardware-based TPM sniffing attacks that require soldering skills and specialized knowledge, Bitpixie enables software-only exploitation that leaves no physical trace.

"The exploitation of the abused Bitpixie vulnerability is non-invasive, does not require any permanent device modifications and no complete disk image, thereby allowing a fast (~5 minutes) compromise," explains researcher Marc Tanner from Compass Security.

This vulnerability specifically targets systems using BitLocker without pre-boot authentication, which is the default configuration on many Windows devices.

Coinbase warns of up to $400 million hit from cyberattack

By **Niket Nishant** and **Chris Prentice**

May 15, 2025 6:59 PM EDT · Updated 7 hours ago

May 15 (Reuters) - Coinbase (COIN.O) forecast a hit of $180 million to $400 million from a cyberattack that breached account data of a "small subset" of its customers, the crypto exchange said in a regulatory filing on Thursday.

The company received an email from an unknown threat actor on May 11, claiming to have information about certain customer accounts as well as internal documents.

While some data — including names, addresses and emails — was stolen, the hackers did not get access to login credentials or passwords, Coinbase said. It would, however, reimburse customers who were tricked into sending funds to the attackers.

Hackers had paid multiple contractors and employees working in support roles outside the U.S. to collect information. The company had fired those involved, it said.

Separately, the U.S. Securities and Exchange Commission had begun scrutinizing whether Coinbase had misstated its user figures, two sources familiar with the matter told Reuters.

Insider Threat

Social Engineering

Millions of RSA encryption keys could be vulnerable to attack

A security flaw in RSA encryption keys makes them easily compromised through a mathematical attack

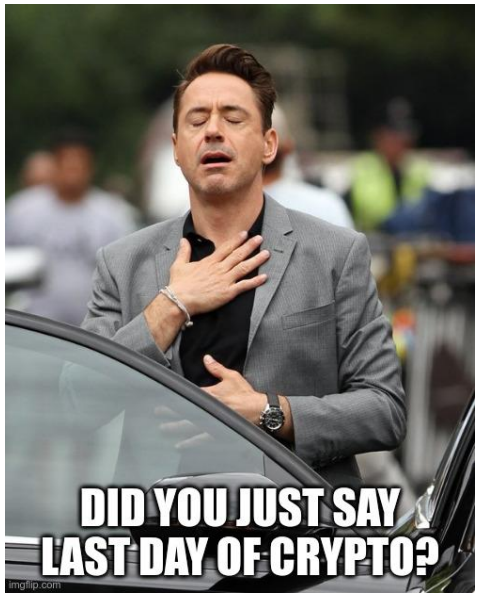
However, according to Keyfactor, this encryption can be cracked. The root of the problem is poor random number generation, with keys sharing prime factors with other keys.

If two keys share a prime factor, both can be broken by computing the Greatest Common Divisor (GCD), allowing the private key to be completely reconstructed.

"This is concerning, as a party with a re-derived private key for an SSL/TLS server certificate can impersonate that entity, and network clients attempting to connect to that endpoint cannot distinguish the attacker from the legitimate holder of the certificate," write Kilgallin and Vasco.

Σήμερα

- Left-over from Hashes
- Authenticated Encryption (AuthEnc)
- Asymmetric/Public Key Cryptography
  - Merkle's Puzzles
  - Diffie-Hellman
  - RSA



Hash Tricks and Datastructures

Application Uses for Hashes

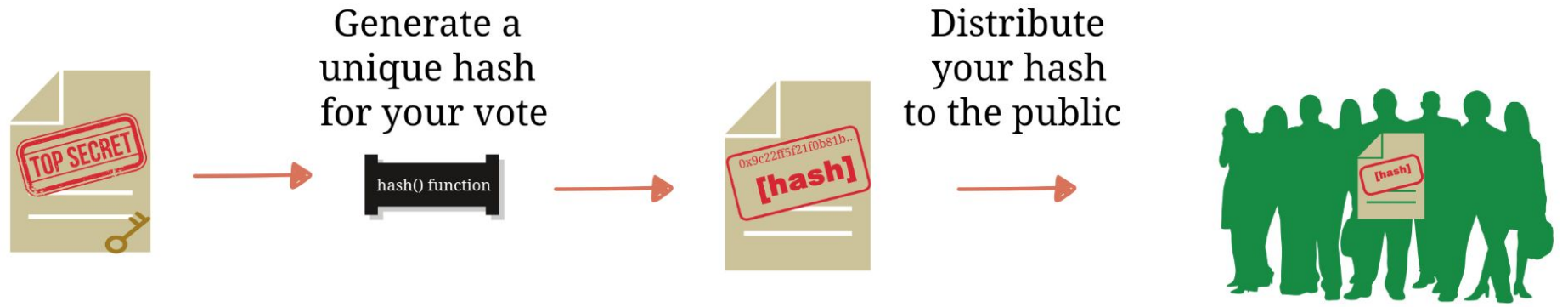
- One-Time Passwords (OTP)
- Data Integrity (tamper-proof)
- Blockchain / micropayments
- Commitment protocols
- and many more

Interview Question #1

You want to prove to another party that you completed an action (e.g., you voted) but not show them your action until after an amount of time has passed (e.g., after election results). How would you implement such a scheme?

Commitment Scheme

1. Commit



Wait for all votes to be committed....

2. Reveal

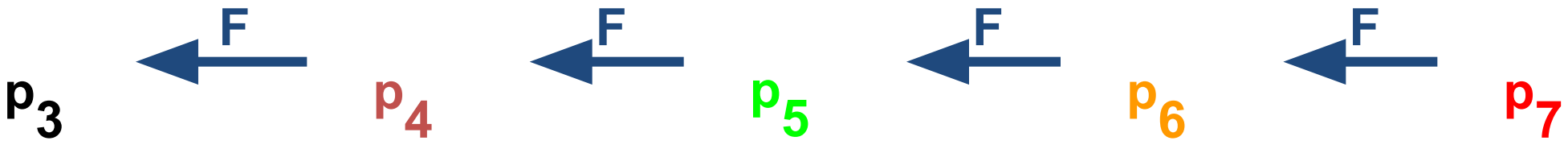
🎉Count votes and declare winner! 🎉

Interview Question #2

Two people want to simulate a fair die roll (1–6) over the internet, but they don't trust each other. How can they do it in such a way that neither party can cheat?

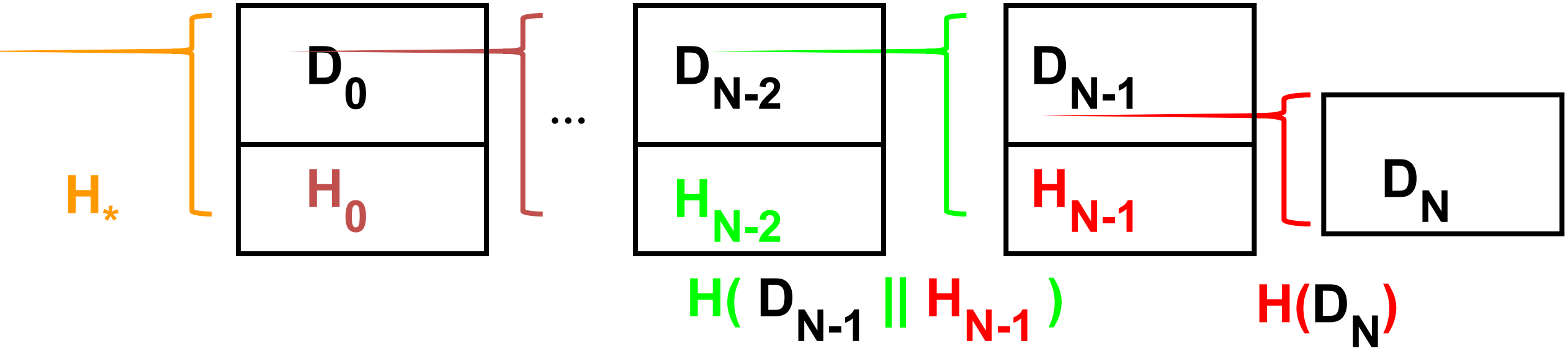
One-Way Chain Application (Lists)

- One-time password system
- Goal
  - Use a different password at every login
  - Server cannot derive password for next login
- Solution: one-way chain
  - Pick random password P_L
  - Prepare sequence of passwords $P_i = F(P_{i+1})$
  - Use passwords $P_0, P_1, \ldots, P_{L-1}, P_L$
  - Server can easily authenticate user



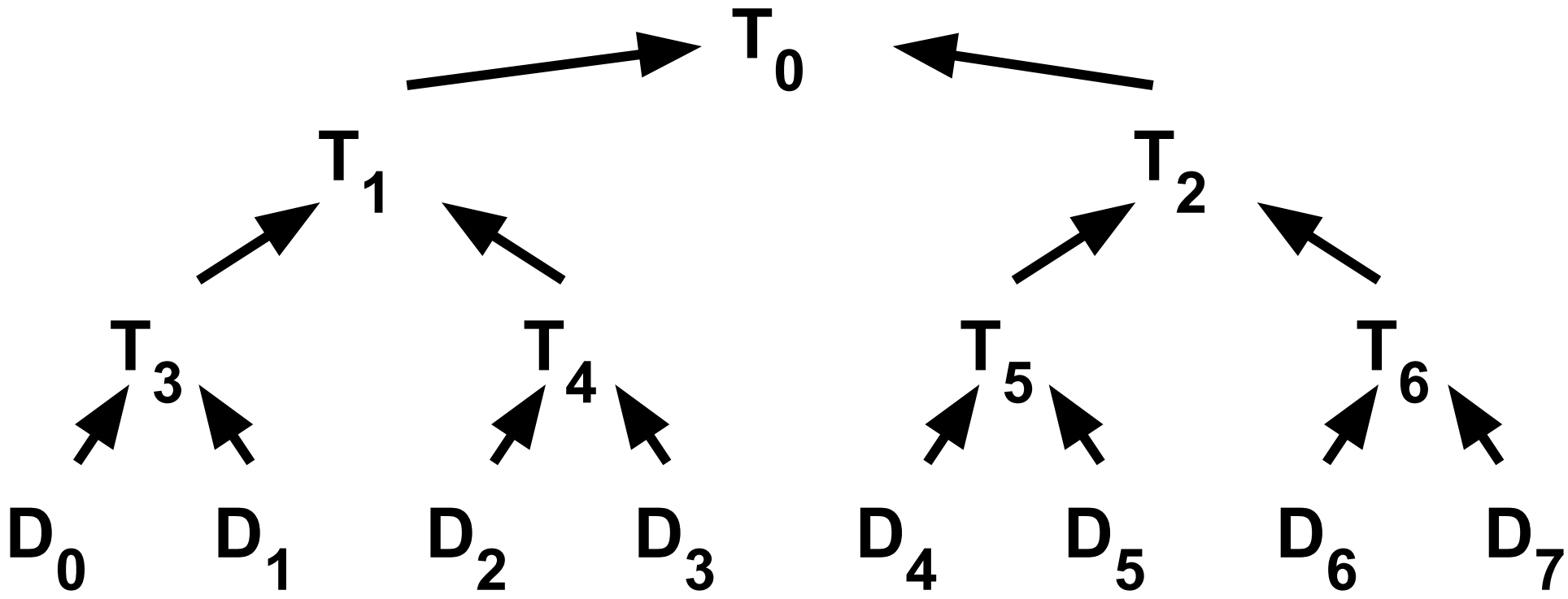
Chained Hashes

- More general construction than one-way hash chains
- Useful for authenticating a sequence of data values D_0 , D_1 , …, D_N
- H_* authenticates entire chain



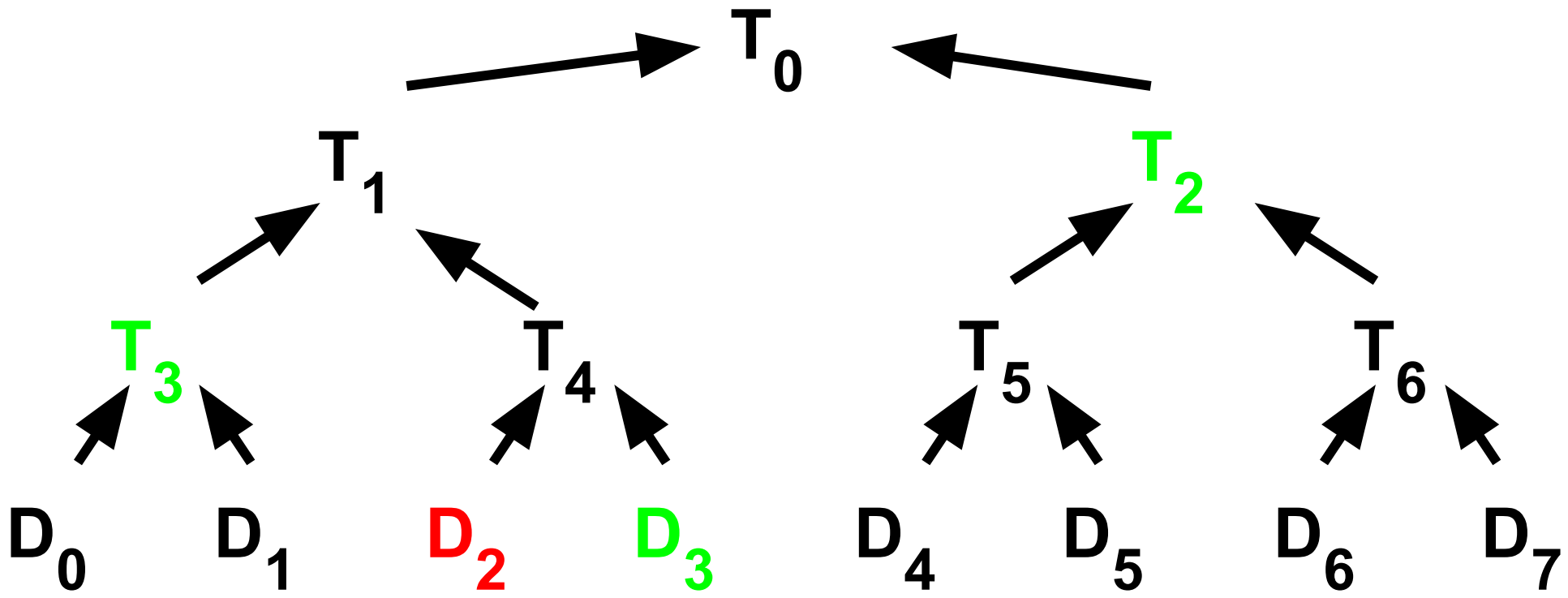
Merkle Hash Trees

- Authenticate a sequence of data values D_0 , D_1 , …, D_N
- Construct binary tree over data values



Merkle Hash Trees II

- Verifier knows T_0
- How can verifier authenticate leaf D_i ?
- Solution: recompute T_0 using D_i
- Example authenticate D_2 , send D_3 T_3 T_2
- Verify $T_0 = H(H(T_3 \| H(D_2 \| D_3)) \| T_2)$




Authenticated
Encryption

Recap: the story so far

Confidentiality: semantic security against a CPA attack

- Encryption secure against **eavesdropping only**

Integrity:

- Existential unforgeability under a chosen message attack
- CBC-MAC, HMAC, *MAC

Can we combine them: encryption secure against **tampering**

- Ensuring both confidentiality and integrity

… but first, some history

Authenticated Encryption (AE): introduced in 2000 [KY'00, BN'00]

Crypto APIs before then: (e.g. MS-CAPI)

- Provide API for CPA-secure encryption (e.g. CBC with rand. IV)
- Provide API for MAC (e.g. HMAC)

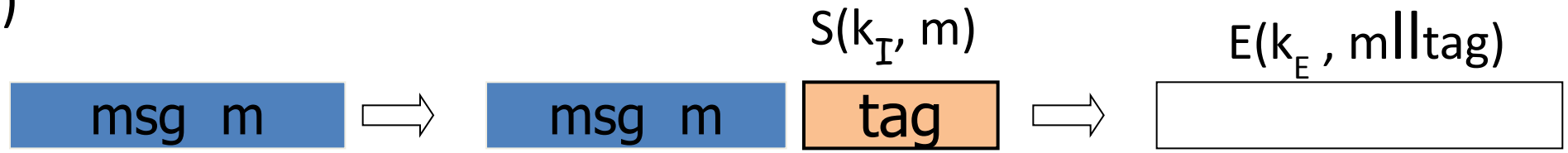
Every project had to combine the two itself without a well defined goal

- Not all combinations provide AE …

Combining MAC and ENC (CCA)

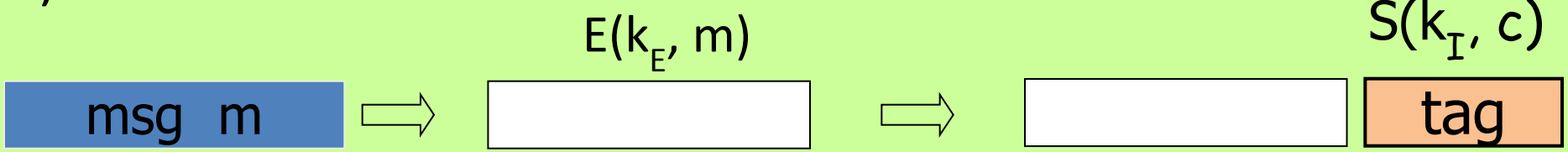
Encryption key k_E. MAC key = k_I

Option 1: (SSL)

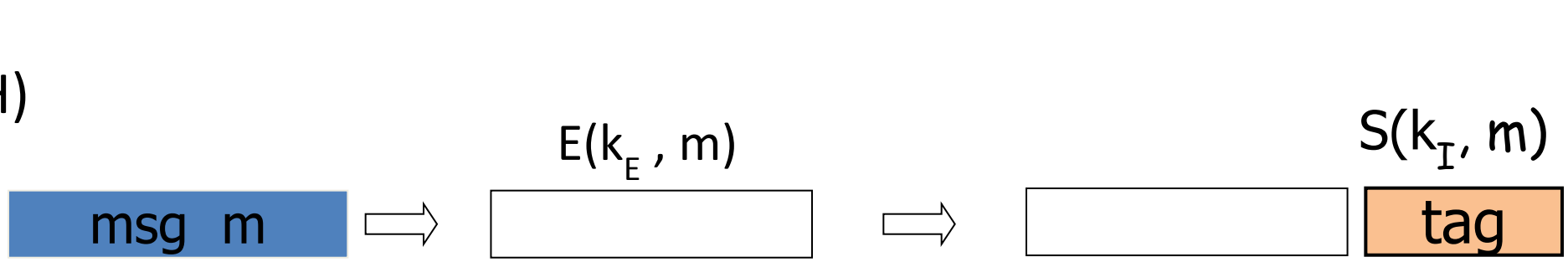


Option 2: (IPsec)

always correct



Option 3: (SSH)



A.E. Theorems

Let (E,D) be CPA secure cipher and (S,V) secure MAC. Then:

1. **Encrypt-then-MAC**: always provides A.E.

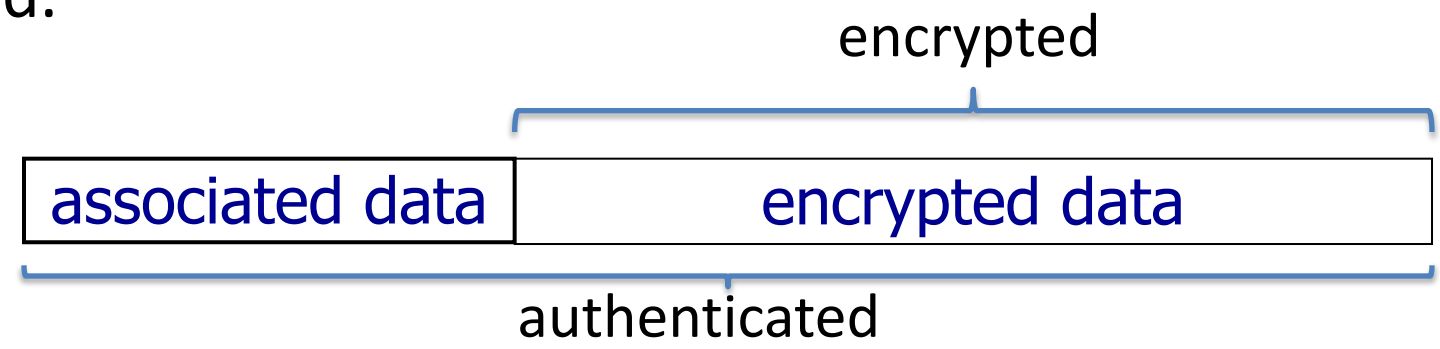
1. **MAC-then-encrypt**: may be insecure against CCA attacks

however: when (E,D) is rand-CTR mode or rand-CBC
M-then-E provides A.E.

Standards (at a high level)

- **GCM**: CTR mode encryption then CW-MAC
  (accelerated via Intel's PCLMULQDQ instruction)
- **CCM**: CBC-MAC then CTR mode encryption (802.11i)
- **EAX**: CTR mode encryption then CMAC

All support AEAD: (auth. enc. with associated data). All are nonce-based.



An example API (OpenSSL)

int **AES_GCM_Init**(AES_GCM_CTX *ain,
    unsigned char ***nonce**, unsigned long noncelen,
    unsigned char ***key**, unsigned int klen)

int **AES_GCM_EncryptUpdate**(AES_GCM_CTX *a,
    unsigned char ***aad**, unsigned long aadlen,
    unsigned char ***data**, unsigned long datalen,
    unsigned char ***out**, unsigned long *outlen)

The TLS Record Protocol (TLS 1.2)

$k_{b \to s}$ ’
$k_{s \to b}$

$k_{b \to s}$ ’
$k_{s \to b}$

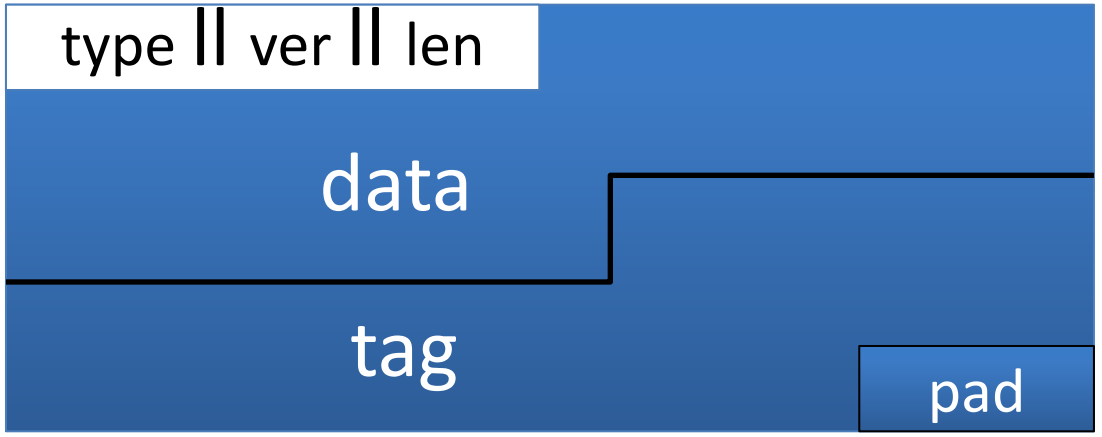
Unidirectional keys: $k_{b \to s}$ and $k_{s \to b}$

Stateful encryption:

- Each side maintains two 64-bit counters: $ctr_{b \to s}$, $ctr_{s \to b}$
- Init. to 0 when session started. ctr++ for every record.
- Purpose: replay defense

TLS record: encryption (CBC AES-128, HMAC-SHA1)

$k_{b \to s} = (k_{mac}, k_{enc})$



Browser side **enc($k_{b \to s}$, data, $ctr_{b \to s}$) :**

step 1: $tag \leftarrow S(k_{mac}, [\ ++ctr_{b \to s} \ ||\ header\ ||\ data])$

step 2: pad [header || data || tag] to AES block size

step 3: CBC encrypt with k_{enc} and new random IV

step 4: prepend header

TLS record: decryption (CBC AES-128, HMAC-SHA1)

Server side **dec($k_{b \to s}$, record, $ctr_{b \to s}$)** :

step 1: CBC decrypt record using k_{enc}

step 2: check pad format: send bad_record_mac if invalid

step 3: check tag on [$++ctr_{b \to s}$ || header || data]

send bad_record_mac if invalid

Provides authenticated encryption

(provided no other info. is leaked during decryption)

Bugs in older versions (prior to TLS 1.1)

IV for CBC is predictable: (chained IV)

IV for next record is last ciphertext block of current record.

Not CPA secure. (a practical exploit: BEAST attack)

Padding oracle: during decryption

if pad is invalid send decryption failed alert

if mac is invalid send bad_record_mac alert

⇒ attacker learns info. about plaintext (various attacks possible)

Lesson: when decryption fails, do not explain why :/

Leaking the length

The TLS header leaks the length of TLS records

- Lengths can also be inferred by observing network traffic

For many web applications, leaking lengths reveals sensitive info:

- In tax preparation sites, lengths indicate the type of return being filed which leaks information about the user's income
- In healthcare sites, lengths leaks what page the user is viewing
- In Google maps, lengths leaks the location being requested

No easy solution

Asymmetric / Public Key Cryptography

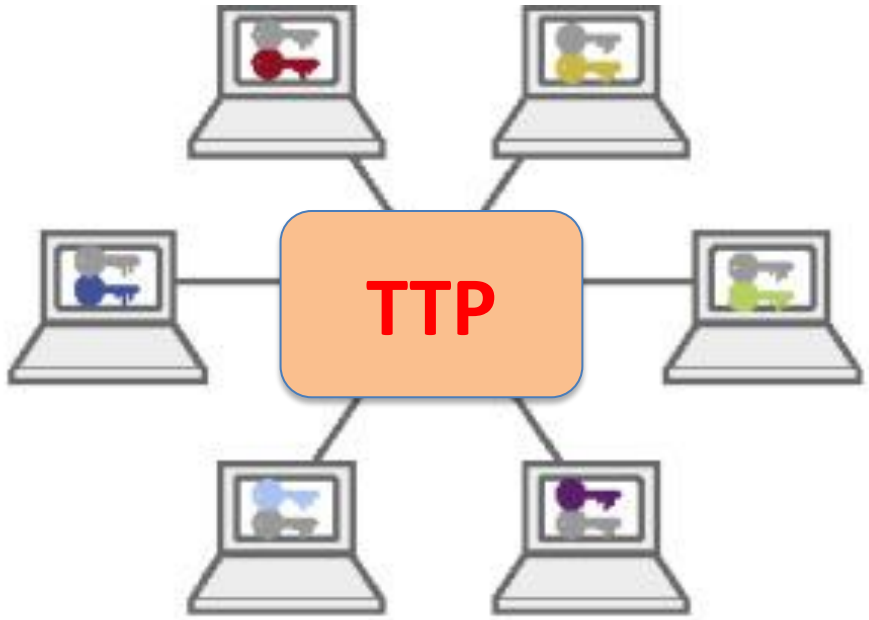
Key management

Problem: n users. Storing mutual secret keys is difficult

Total: O(n) keys per user

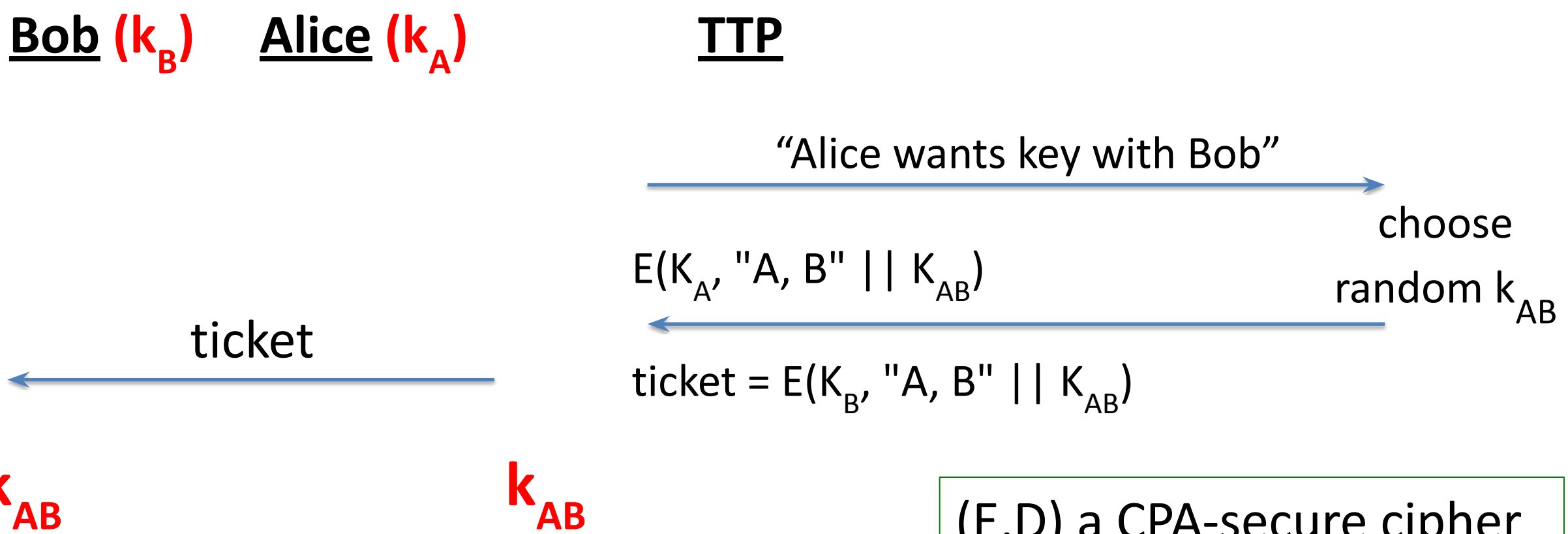
A better solution

Online Trusted 3rd Party (TTP)



Generating keys: a toy protocol

Alice wants a shared key with Bob. Eavesdropping security only.



(E,D) a CPA-secure cipher

Generating keys: a toy protocol

Alice wants a shared key with Bob. Eavesdropping security only.

Eavesdropper sees: $E(k_A,\ \text{“A, B”} \,||\, k_{AB})$; $E(k_B,\ \text{“A, B”} \,||\, k_{AB})$

(E,D) is CPA-secure $\Rightarrow$
eavesdropper learns nothing about k_{AB}

Note: TTP needed for every key exchange, knows all session keys.

(basis of Kerberos system)

Toy protocol: insecure against active attacks

Example: insecure against replay attacks

Attacker records session between Alice and merchant Bob

- For example a book order

Attacker replays session to Bob

- Bob thinks Alice is ordering another copy of book

Key question

Can we generate shared keys without an **online** trusted 3rd party?

Answer: yes!

Starting point of public-key cryptography:

- Merkle (1974), Diffie-Hellman (1976), RSA (1977)

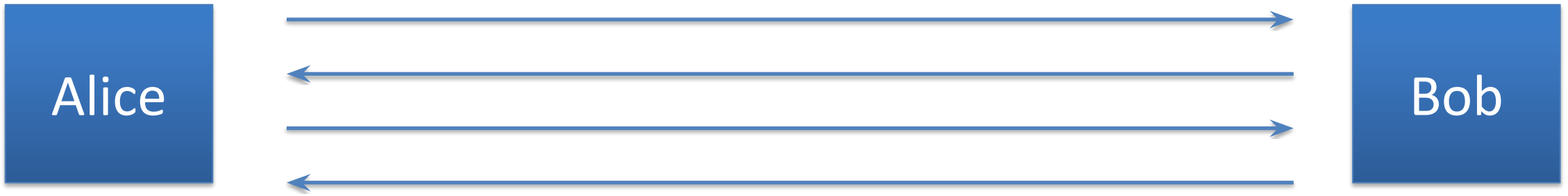
- More recently: ID-based enc. (BF 2001), Functional enc. (BSW 2011)

Merkle Puzzles

Key exchange without an online TTP?

Goal: Alice and Bob want shared key, unknown to eavesdropper

- For now: security against eavesdropping only (no tampering)



eavesdropper ??

Can this be done using generic symmetric crypto?

Merkle Puzzles (1974)

Answer: yes, but very inefficient

Main tool: puzzles

- Problems that can be solved with some effort
- Example: E(k,m) a symmetric cipher with $k \in \{0,1\}^{128}$
  - **puzzle(P) = E(P, "message")** where $P = 0^{96} \,||\, b_1 \ldots b_{32}$
  - Goal: find P by trying all 2^{32} possibilities

Merkle puzzles

Alice: prepare 2^{32} puzzles

- For i=1, …, 2^{32} choose random $\mathbf{P_i} \in \mathbf{\{0,1\}^{32}}$ and $\mathbf{x_i, k_i} \in \mathbf{\{0,1\}^{128}}$

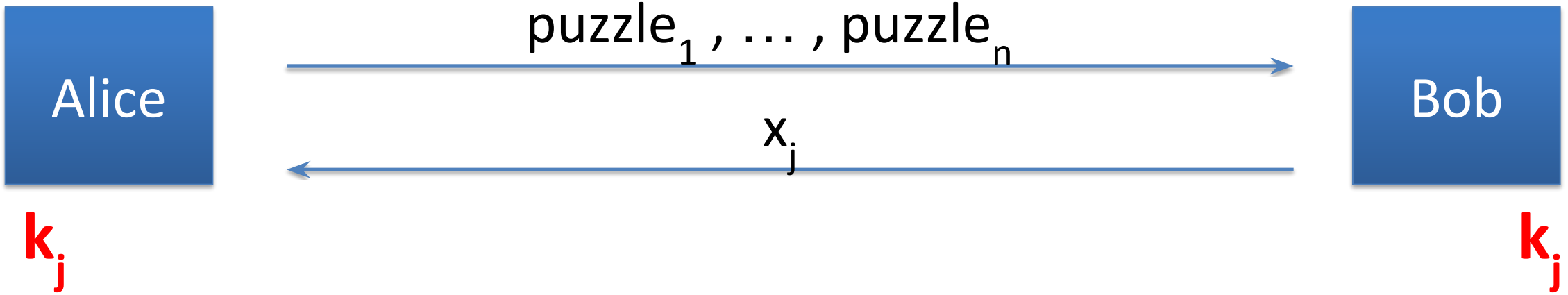
  set $\text{puzzle}_i \leftarrow E(0^{96} \,||\, \mathbf{P_i}, \text{“Puzzle \# } x_i \text{”} \,||\, k_i)$
- Send $\text{puzzle}_1, \ldots, \text{puzzle}_{2\text{^}32}$ to Bob

Bob: choose a random puzzle_j and solve it. Obtain (x_j, k_j).

- Send x_j to Alice

Alice: lookup puzzle with number x_j. Use k_j as shared secret

In a figure



Alice's work: O(n) (prepare n puzzles)

Bob's work: O(n) (solve one puzzle)

Eavesdropper's work:  (e.g. 2^{64} time)

Impossibility Result

Can we achieve a better gap using a general symmetric cipher?

Answer: unknown

But: roughly speaking,

quadratic gap is best possible if we treat cipher as

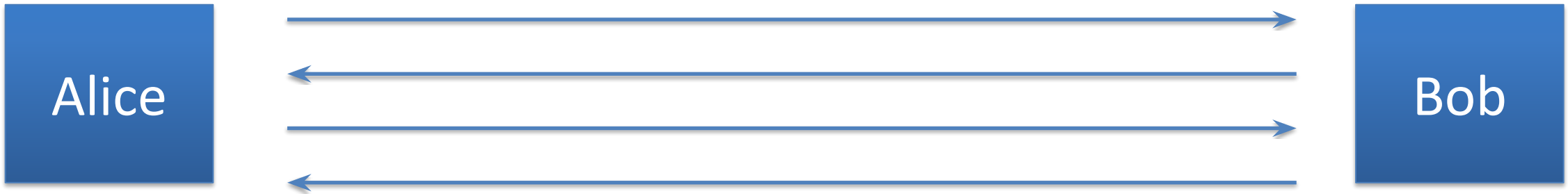
a black box oracle [IR'89, BM'09]

The Diffie-Hellman (DH) Protocol

Key exchange without an online TTP?

Goal: Alice and Bob want shared secret, unknown to eavesdropper

- For now: security against eavesdropping only (no tampering)



eavesdropper ??

Can this be done with an exponential gap?

The Diffie-Hellman protocol (informally)

Fix a large prime p (e.g. 600 digits)

Fix an integer g in {1, …, p}

Alice **Bob**

choose random **a** in {1,…,p-1} choose random **b** in {1,…,p-1}

$A = g^a \pmod p$

$B = g^b \pmod p$

$\mathbf{B^a} \pmod p = (g^b)^a = \mathbf{k_{AB} = g^{ab}} \pmod p$

Security (much more on this later)

Eavesdropper sees: p, g, $A=g^a \pmod p$, and $B=g^b \pmod p$

Can she compute $g^{ab} \pmod p$??

More generally: define $DH_g(g^a, g^b) = g^{ab} \pmod p$

How hard is the DH function mod p?

How hard is the DH function mod p?

Suppose prime p is n bits long.

Best known algorithm (GNFS): run time exp($\tilde{O}(\sqrt[3]{n})$)

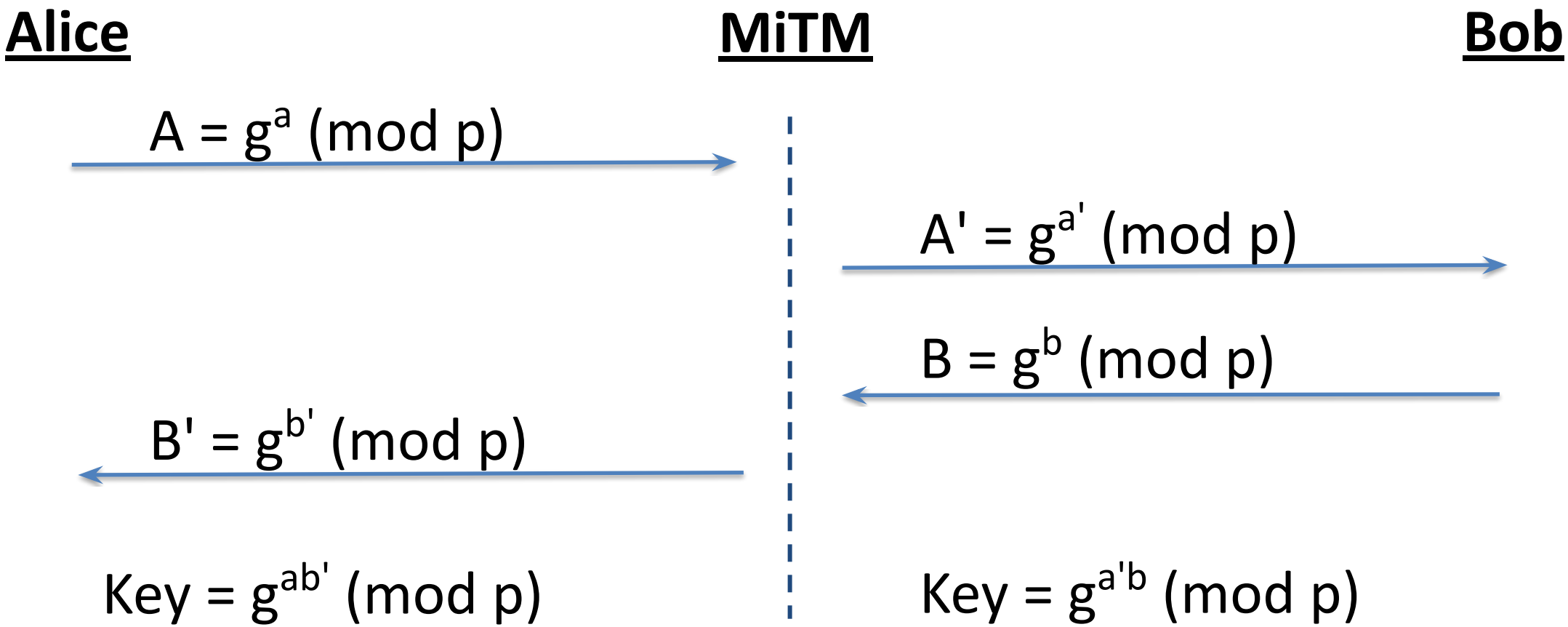
cipher key size	modulus size	Elliptic Curve size
80 bits	1024 bits	160 bits
128 bits	3072 bits	256 bits
256 bits (AES)	**15360** bits	512 bits

As a result: slow transition away from (mod p) to elliptic curves

Elliptic curve
Diffie-Hellman

Insecure against man-in-the-middle

As described, the protocol is insecure against **active** attacks

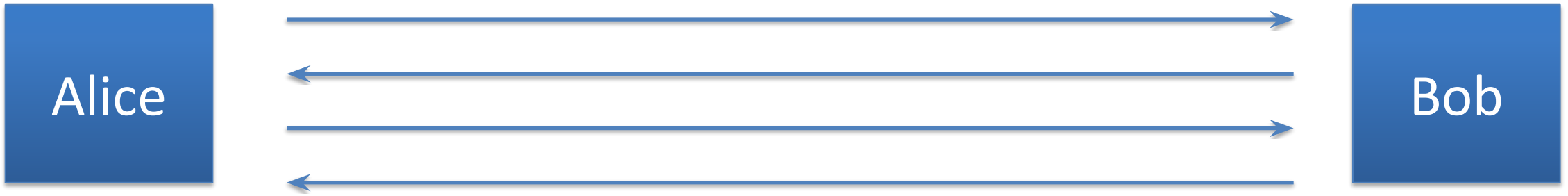


Public Key Cryptography

Establishing a shared secret

Goal: Alice and Bob want shared secret, unknown to eavesdropper

- For now: security against eavesdropping only (no tampering)



eavesdropper ??

This segment: a different approach

Public key encryption

Def: a public-key encryption system is a triple of algs. (G, E, D)

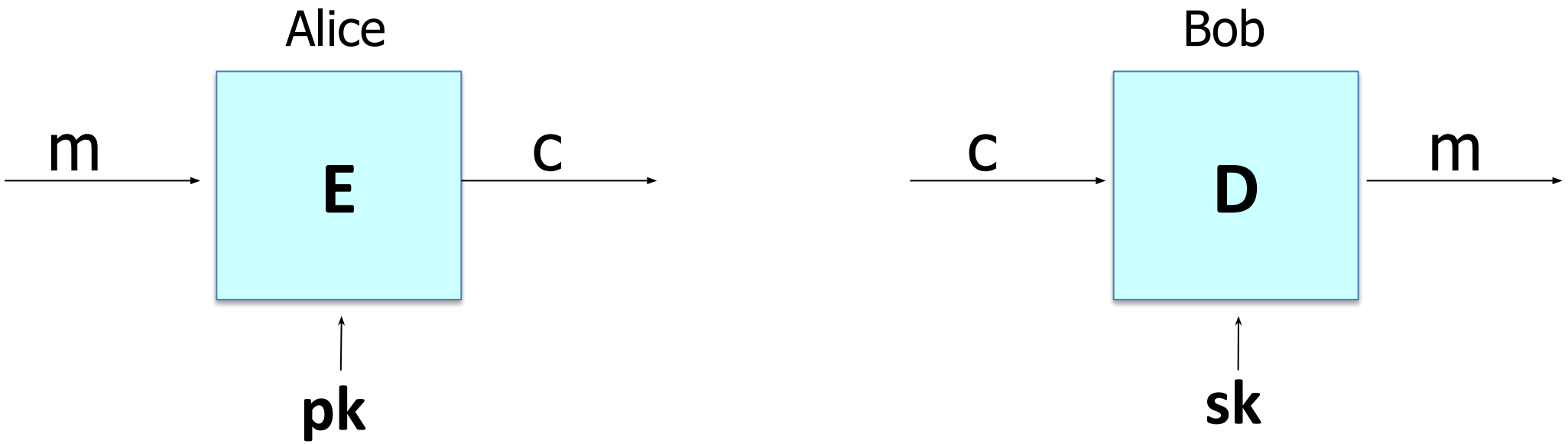
- G(): randomized alg. outputs a key pair (pk, sk)
- E(pk, m): randomized alg. that takes $m \in M$ and outputs $c \in C$
- D(sk,c): det. alg. that takes $c \in C$ and outputs $m \in M$ or $\perp$

Consistency: $\forall$ (pk, sk) output by G :

$\forall m \in M$: D(sk, E(pk, m)) = m

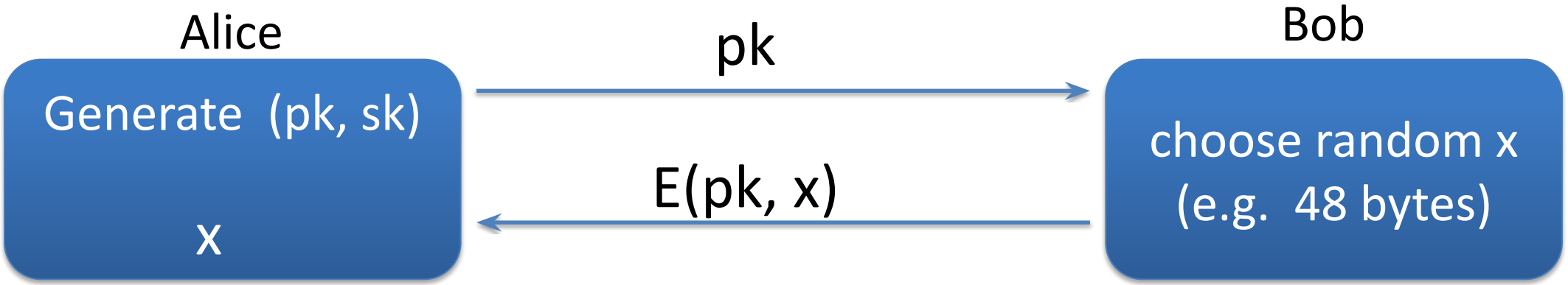
Public key encryption

Bob: generates (PK, SK) and gives PK to Alice



Applications

Session setup (for now, only eavesdropping security)



Non-interactive applications: (e.g. Email)

- Bob sends email to Alice encrypted using pk_{alice}
- Note: Bob needs pk_{alice} (public key management)

Trapdoor functions (TDF)

Def: a trapdoor func. $X \longrightarrow Y$ is a triple of efficient algs. (G, F, F^{-1})

- G(): randomized alg. outputs a key pair (pk, sk)
- F(pk, $\cdot$): det. alg. that defines a function $X \longrightarrow Y$
- F^{-1}(sk, $\cdot$): defines a function $Y \longrightarrow X$ that inverts F(pk, $\cdot$)

More precisely: $\forall$ (pk, sk) output by G

$$\forall x \in X: \quad F^{-1}(sk, F(pk, x)) = x$$

The RSA trapdoor permutation

First published: Scientific American, Aug. 1977.

Very widely used:

- SSL/TLS: certificates and key-exchange
- Secure e-mail and file systems

… many others

The RSA trapdoor permutation

G(): choose random primes p,q ≈1024 bits. Set **N=pq**.

choose integers **e , d** s.t. $\mathbf{e \cdot d = 1 \pmod{\phi(N)}}$ **where** $\mathbf{\phi(N) = (p - 1)(q - 1)}$

output pk = (N, e) , sk = (N, d)

F(pk, x): $\mathbb{Z}_N^* \rightarrow \mathbb{Z}_N^*$; **RSA(x) = x^e** (in Z_N)

F^{-1}(sk, y) = y^d ; y^d = **RSA(x)^d** = x^{ed} = $x^{k\phi(N)+1}$ = $\left(x^{\phi(N)}\right)^k \cdot$ **x** = x

The RSA Assumption

RSA assumption: RSA is one-way permutation

For all efficient algs. A:

$$\Pr\left[\ A(N,e,y) = y^{1/e}\ \right] < \text{negligible}$$

where $p,q \xleftarrow{R}$ n-bit primes, $N \leftarrow pq$, $y \xleftarrow{R} Z_N^*$

Review: RSA pub-key encryption (ISO std)

(E_s, D_s): symmetric enc. scheme providing auth. encryption.

H: $Z_N \rightarrow K$ where K is key space of (E_s, D_s)

- **G**(): generate RSA params: pk = (N,e), sk = (N,d)
- **E**(pk, m): (1) choose random x in Z_N

  (2) $y \leftarrow RSA(x) = x^e$, $k \leftarrow H(x)$

  (3) output $(y , E_s(k,m))$
- **D**(sk, (y, c)): output $D_s(H(RSA^{-1}(y)) , c)$

Textbook RSA is insecure

Textbook RSA encryption:

- public key: **(N,e)** Encrypt: $\mathbf{c \leftarrow m^e}$ (in Z_N)
- secret key: **(N,d)** Decrypt: $\mathbf{c^d \rightarrow m}$

Insecure cryptosystem !!

- Is not semantically secure and many attacks exist

⇒ The RSA trapdoor permutation is not an encryption scheme !

Ευχαριστώ και καλή μέρα εύχομαι!

Keep hacking!