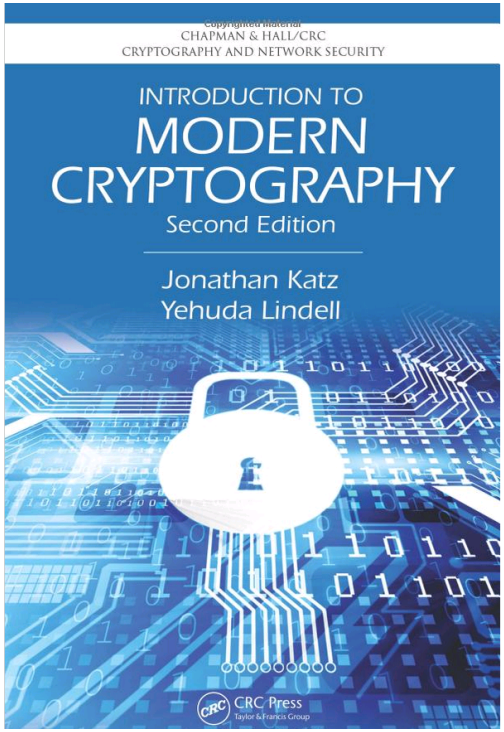
Public Key Cryptography

Ali El Kaafarani

[1]Mathematical Institute
[2] PQShield Ltd.

Outline

Course main reference



Prime numbers

- Any integer n can be decomposed uniquely as a product of prime numbers.
- There are infinitely many primes.
- **Prime Number Theorem**: the number of primes up to some bound B is roughly equal to $B/\log B$.
- **Bertrand's postulate**: For any $n > 1$, the fraction of the n-bit integers that are prime is at least $1/3n$.

Primality testing

- Given an integer n, decide whether n is prime or not.
- There are deterministic algorithms for primality testing (see AKS test).
- In practice, we use probabilistic algorithms (having a small probability to return prime for composite numbers) that are much faster.

Generating Random Primes

You can generate primes by picking random numbers smaller than B and checking whether they are prime: need about $\log B$ trials by the prime number theorem. More formally,

Algorithm

Input: Length n*, parameter* t
For $i = 1$ *to* t*:*
  $p' \leftarrow \{0,1\}^{n-1}$
  $p := 1||p'$
  if *Primality_test* $(p) = 1$ ***return*** p
return *fail*

Generating Random Primes

- Remember that for any $n > 1$, the fraction of the n-bit integers that are prime is at least $1/3n$.
- Now, set $t = 3n^2$, then the probability of the previous algorithm to not output a prime in t iteration is

$$(1 - \frac{1}{3n})^t = \left((1 - \frac{1}{3n})^{3n}\right)^n \leq (e^{-1})^n = e^{-n}$$

- This probability is negligible in n if we have a number of iterations that is polynomial in n.
- We still need to study the algorithms that test numbers primality!

Fermat test

- Observation: if n is prime then $a^{n-1} = 1 \bmod n$ for all a (Fermat's little theorem)
- Idea: choose random a and check whether $a^{n-1} = 1 \bmod n$. If not then n is composite.
- We define a *witness* that n is composite any $a \in \mathbb{Z}_n^*$, s.t. $a^{n-1} \neq 1 \mod n$.

Fermat test

Algorithm

Input: Integer n *and parameter* 1^t

for $i = 1$ *to* t

$a \leftarrow \{1, \cdots, n-1\}$

if $a^{n-1} \neq 1 \mod n$ ***return*** *"composite"*

return *"prime"*

Theorem

If n *has a witness that it is composite, then* $|\{\text{witnesses}\}_n| \geq |\mathbb{Z}_n^*|/2$

Fermat test

Algorithm

Input: Integer n *and parameter* 1^t
for $i = 1$ *to* t
  $a \leftarrow \{1, \cdots, n-1\}$
  if $a^{n-1} \neq 1 \mod n$ ***return*** *"composite"*
return *"prime"*

Theorem

If n *has a witness that it is composite, then* $|\{\text{witnesses}\}_n| \geq |\mathbb{Z}_n^*|/2$

However, try 561 or 41041.

Fermat test

Algorithm

Input: Integer n *and parameter* 1^t
for $i = 1$ *to* t
  $a \leftarrow \{1, \cdots, n-1\}$
  if $a^{n-1} \neq 1 \mod n$ ***return*** *"composite"*
return *"prime"*

Theorem

If n *has a witness that it is composite, then* $|\{\text{witnesses}\}_n| \geq |\mathbb{Z}_n^*|/2$

However, try 561 or 41041.
Carmichael numbers: are composite and pass this test for all $0 < a < n$, i.e. they don't have any witnesses.

Fermat test

Algorithm

Input: Integer n *and parameter* 1^t
for $i = 1$ *to* t
  $a \leftarrow \{1, \cdots, n-1\}$
  if $a^{n-1} \neq 1 \mod n$ ***return*** *"composite"*
return *"prime"*

Theorem

If n *has a witness that it is composite, then* $|\{\text{witnesses}\}_n| \geq |\mathbb{Z}_n^*|/2$

However, try 561 or 41041.

Carmichael numbers: are composite and pass this test for all $0 < a < n$, i.e. they don't have any witnesses.

Solution?

Testing Primality

- We have just seen that Carmichael numbers don't have any witnesses!
- We need to refine Fermat's test.
- Let $n - 1 = 2^k u$, where u is odd and $k \geq 1$ (and therefore n is odd).

Testing Primality

- We have just seen that Carmichael numbers don't have any witnesses!
- We need to refine Fermat's test.
- Let $n - 1 = 2^k u$, where u is odd and $k \geq 1$ (and therefore n is odd).
- In Fermat's test, we check if $a^{n-1} = a^{2^k u} = 1$.

Testing Primality

- We have just seen that Carmichael numbers don't have any witnesses!
- We need to refine Fermat's test.
- Let $n - 1 = 2^k u$, where u is odd and $k \geq 1$ (and therefore n is odd).
- In Fermat's test, we check if $a^{n-1} = a^{2^k u} = 1$.
- What about $a^u, a^{2u}, \cdots, a^{2^{k-1}u}$?

Testing Primality

- We have just seen that Carmichael numbers don't have any witnesses!
- We need to refine Fermat's test.
- Let $n-1 = 2^k u$, where u is odd and $k \geq 1$ (and therefore n is odd).
- In Fermat's test, we check if $a^{n-1} = a^{2^k u} = 1$.
- What about $a^u, a^{2u}, \cdots, a^{2^{k-1}u}$?
- **Strong witness:** $a \in \mathbb{Z}_n^*$ is a strong witness that n is composite if
  - $a^u \neq \pm 1 \mod n$ and
  - $a^{2^i u} \neq -1$ for all $i \in \{1, \cdots, k-1\}$

Testing Primality

- If n is prime, then n doesn't have any strong witness that it is composite. More formally,

Theorem

Let n be an odd number that is not a prime power, then we have that at least half of the elements of $\mathbb{Z}_n^*$ are strong witnesses that n is composite.

Miller-Rabin test

Algorithm

Input: Integer $n > 2$ *and parameter* 1^t
If n *is even,* ***return*** *"composite"*
If n *is a perfect power,* ***return*** *"composite"* [a]
Write $n - 1 = 2^k u$ *where* u *is odd and* $k \geq 1$
for $j = 1$ *to* t
  $a \leftarrow \{1, \cdots, n-1\}$
  if $a^u \neq \pm 1 \mod n$ ***and*** $a^{2^i u} \neq -1 \mod n$ ***for*** $i \in \{1, \cdots, k-1\}$
    return *"composite"*
return *"prime"*

[a] Exercise: Show that this test can be done in polynomial time

Miller-Rabin test

Theorem

If p is prime, then the Miller-Rabin test always outputs "prime. If p is composite, the algorithm outputs "composite" except with probability at most 2^{-t}

Miller-Rabin test

Theorem

If p is prime, then the Miller-Rabin test always outputs "prime. If p is composite, the algorithm outputs "composite" except with probability at most 2^{-t}

(Exercise-1) Show that the Miller-Rabin algorithm runs in time polynomial in $|p|$ and t.

Miller-Rabin test

Theorem

If p is prime, then the Miller-Rabin test always outputs "prime. If p is composite, the algorithm outputs "composite" except with probability at most 2^{-t}

(Exercise-1) Show that the Miller-Rabin algorithm runs in time polynomial in $|p|$ and t.
(Exercise-2) Compare its running time to the (deterministic) AKS' running time.

Outline

Public Key Cryptosystems

An asymmetric encryption scheme consists of the following algorithms:

- KeyGen(1^n): is a randomized algorithm that takes the security parameters as input and returns a pair of keys $(\mathsf{PK}, \mathsf{SK})$, the public key PK and its matching secret key SK, respectively.
- Enc(PK, m): A randomized algorithm that takes a public key PK, a plaintext m and returns a ciphertext c.
- Dec(SK, c): A deterministic algorithm that takes the secret key SK and a ciphertext c, and returns a message $m \in \mathcal{M} \cup \perp$.

Correctness:

$$\forall m \in \mathcal{M}, \Pr[(\mathsf{SK}, \mathsf{PK}) \leftarrow \mathsf{KeyGen}(n) : \mathsf{Dec}(\mathsf{Enc}(\mathsf{PK}, m), \mathsf{SK}) = m] = 1$$

CPA Indistinguishability Experiment $\mathsf{PubK}^{\mathsf{cpa}}_{\mathcal{A},E}$

Challenger Ch | **Adversary** A

$\mathsf{PK}, \mathsf{SK} \leftarrow \mathsf{KeyGen}(n)$

$\xleftarrow{m_0, m_1, |m_0| = |m_1|}$

$b \leftarrow \{0, 1\}$

$\xrightarrow{c = \mathsf{Enc}(\mathsf{PK}, m_b)}$ Outputs his guess b'

Definition

An encryption scheme is CPA-secure if for all efficient $\mathcal{A}$ the following holds:

$$\mathsf{Adv}^{\mathsf{cpa}}_{\mathcal{A},E}(n) = \Pr[\mathsf{PubK}^{\mathsf{cpa}}_{\mathcal{A},E}(n) = 1] - 1/2 = \mathsf{negl}(n)$$

Where $\mathsf{PubK}^{\mathsf{cpa}}_{\mathcal{A},E}(n) = 1$ if $b' = b$, and 0 otherwise.

CCA Indistinguishability Experiment $\mathsf{PubK}^{\mathsf{cca}}_{\mathcal{A},E}$

Challenger Ch		**Adversary** A
$\mathsf{PK}, \mathsf{SK} = \mathsf{KeyGen}(n)$		Access to the oracle $\mathsf{Dec}(\mathsf{SK}, \cdot)$
	$\xleftarrow{m_0, m_1, \lvert m_0\rvert = \lvert m_1\rvert}$	
$b \leftarrow \{0,1\}$		
	$\xrightarrow{c = \mathsf{Enc}(\mathsf{PK}, m_b)}$	Access to the oracle $\mathsf{Dec}(\mathsf{SK}, \cdot)^c$
		Outputs his guess b'

Definition

An encryption scheme is CCA-secure if for all efficient $\mathcal{A}$ the following holds:

$$\mathsf{Adv}^{\mathsf{cca}}_{\mathcal{A},E}(n) = \Pr[\mathsf{PubK}^{\mathsf{cca}}_{\mathcal{A},E}(n) = 1] - 1/2 = \mathsf{negl}(n)$$

Dealing with arbitrary-length messages

Theorem

If a public-key encryption scheme is CPA-secure, then it also has indistinguishable multiple encryptions, where the adversary is allowed to send two lists of messages to be challenged on instead of sending a pair of messages.

- As a consequence, any CPA-secure public-key encryption scheme for fixed-length messages (down to one bit!) can be used as a public key-encryption scheme for arbitrary-length messages.

Hybrid Encryption

- A better approach to deal with arbitrary-length messages.
- We will use a private-key encryption scheme along with a public-key encryption scheme.
- Remember that private-key encryption scheme are significantly faster than public ones.
- We call this approach the *key-encapsulation mechanism* and *data-encapsulation mechanism* (KEM/DEM).

KEM

An key-encapsulation mechanism scheme (KEM) consists of the following PPT algorithms:

- KeyGen(1^n): takes the security parameter as input and returns a pair of keys $(\mathsf{PK}, \mathsf{SK})$, the public key PK and its matching secret key SK, respectively, each of length n.
- Encaps$(\mathsf{PK}, 1^n)$: it returns a ciphertext c and a key $k \in \{0,1\}^{\ell(n)}$.
- Decaps(SK, c): A deterministic algorithm that takes a secret key SK and a ciphertext c, and returns a key k or $\perp$.

Hybrid Encryption

An hybrid encryption scheme consists of a KEM scheme and a private-key encryption scheme:

- $\text{KeyGen}^{hy}(1^n)$: is a randomized algorithm that takes the security parameters as input and returns a pair of keys (PK, SK).
- $\text{Enc}^{hy}(\text{PK}, m \in \{0,1\}^*)$: takes a public key PK, a plaintext m and does the following:
  - compute $(c, k) \leftarrow \text{Encaps}(\text{PK}, 1^n)$.
  - compute $c' \leftarrow \text{Enc}(k, m)$.
  - output the ciphertext (c, c').
- $\text{Dec}^{hy}(\text{SK}, (c, c'))$: takes a secret key SK and a ciphertext (c, c') and does the following:
  - $k \leftarrow \text{Decaps}(\text{SK}, c)$.
  - output $m \leftarrow \text{Dec}(k, c')$.

Hybrid Encryption: Efficiency

- Fix n. Let $\alpha =$cost(Encaps(1^n)) and $\beta =$cost(Enc$(1$ bit$)$). Then

$$\text{cost}(\text{Enc}^{hy}(1 \text{ bit})) = \frac{\alpha + \beta \cdot |m|}{|m|} = \frac{\alpha}{|m|} + \beta$$

- For sufficiently large m, cost$(\text{Enc}^{hy}(1 \text{ bit})) \to \beta$. In other words, cost$(\text{Enc}^{hy}(1 \text{ bit})) \approx$ cost$(\text{Enc}(1 \text{ bit}))$, which is the cost of encrypting one bit using a private-key encryption scheme!

Security of KEM

Intuitively speaking, for a KEM to be CPA secure, we require the encapsulated key to be indistinguishable from a uniform key that is independent of the ciphertext.

Experiment

- *Run* KeyGen(1^n) *to get* $(\mathsf{PK}, \mathsf{SK})$*, then run* Encaps$(\mathsf{PK}, 1^n)$ *to generate* (c, k) *where* $k \in \{0,1\}^n$*.*
- *Choose random* $b \in \{0,1\}$*, if* $b = 0$ *set* $\bar{k} := k$*, otherwise choose* $\bar{k}$ *uniformly at random from* $\{0,1\}^n$*.*
- *Give the adversary* $\mathcal{A}$ *the tuple* $(\mathsf{PK}, c, \bar{k})$*, he should output a bit* b'*.*
- *Experiment output: 1 if* $b' = b$ *and 0 otherwise.*

Security of the Hyprid Encryption Scheme

Theorem

The hybrid encryption scheme is a CPA-secure public-key encryption scheme if KEM is CPA secure and the private-key encryption scheme has indistinguishable encryptions in the presence of an eavesdropper.

Security of the Hyprid Encryption Scheme

Let $k \leftarrow \text{Encaps}(1^n)$ and $k' \leftarrow \{0,1\}^n$

$(pk, Encaps(pk, 1^n), Enc(k', m_0))$

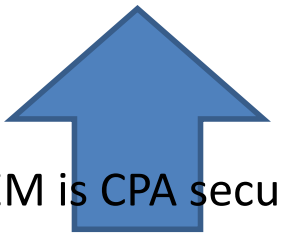
$(pk, Encaps(pk, 1^n), Enc(k', m_1))$

Security of the Hyprid Encryption Scheme

$(pk, Encaps(pk, 1^n), Enc(k', m_0))$ $\longleftrightarrow$ $(pk, Encaps(pk, 1^n), Enc(k', m_1))$

PrivEnc is secure

Security of the Hyprid Encryption Scheme

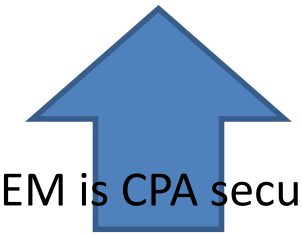


$(pk, Encaps(pk, 1^n), Enc(k', m_0))$ $\longleftrightarrow$ $(pk, Encaps(pk, 1^n), Enc(k', m_1))$

PrivEnc is secure

Security of the Hyprid Encryption Scheme

$(pk, Encaps(pk, 1^n), Enc(k, m_0))$



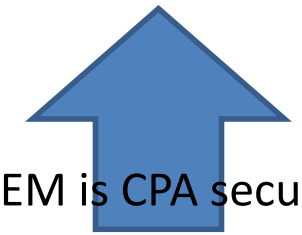
KEM is CPA secure

$(pk, Encaps(pk, 1^n), Enc(k', m_0))$ $\longleftrightarrow$ $(pk, Encaps(pk, 1^n), Enc(k', m_1))$

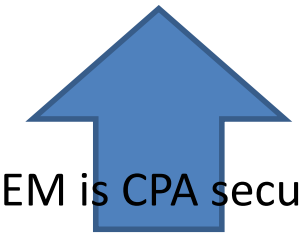
PrivEnc is secure

Security of the Hyprid Encryption Scheme

$(pk, Encaps(pk, 1^n), Enc(k, m_0))$



KEM is CPA secure



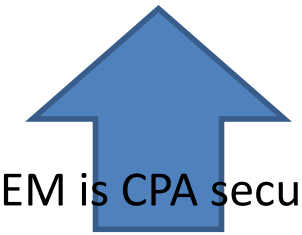
KEM is CPA secure

$(pk, Encaps(pk, 1^n), Enc(k', m_0))$ $\longleftrightarrow$ $(pk, Encaps(pk, 1^n), Enc(k', m_1))$

PrivEnc is secure

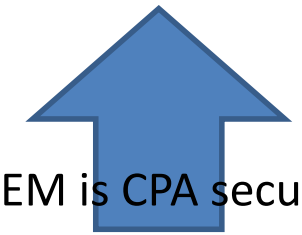
Security of the Hyprid Encryption Scheme

$(pk, Encaps(pk, 1^n), Enc(k, m_0))$



KEM is CPA secure

$(pk, Encaps(pk, 1^n), Enc(k, m_1))$



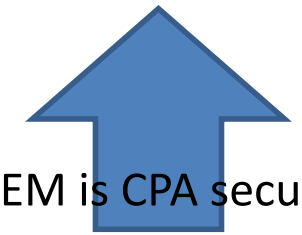
KEM is CPA secure

$(pk, Encaps(pk, 1^n), Enc(k', m_0))$ $\longleftrightarrow$ $(pk, Encaps(pk, 1^n), Enc(k', m_1))$

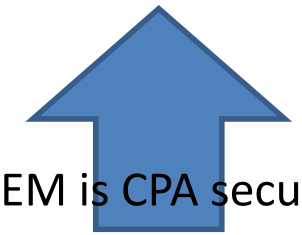
PrivEnc is secure

Security of the Hyprid Encryption Scheme

$(pk, Encaps(pk, 1^n), Enc(k, m_0)) \approx (pk, Encaps(pk, 1^n), Enc(k, m_1))$



KEM is CPA secure



KEM is CPA secure

$(pk, Encaps(pk, 1^n), Enc(k', m_0)) \longleftrightarrow (pk, Encaps(pk, 1^n), Enc(k', m_1))$

PrivEnc is secure

Security of the Hyprid Encryption Scheme

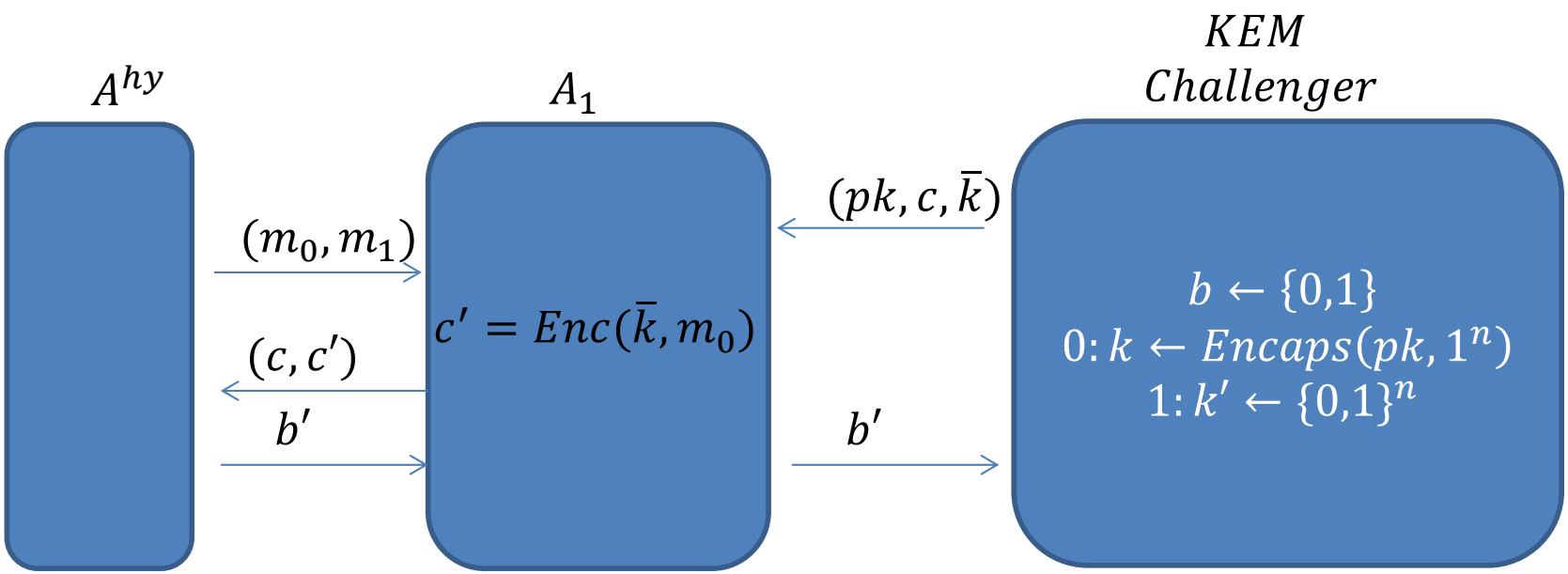
We need to prove the following:

$$\Pr[\mathsf{PubK}^{eav}_{\mathcal{A}^{hy},S^{hy}}(n) = 1] \leq \frac{1}{2} + \mathsf{negl}(n)$$

Whereas, by definition of the security experiment, we have

$$\begin{aligned}\Pr[\mathsf{PubK}^{eav}_{\mathcal{A}^{hy},S^{hy}}(n) = 1] = \frac{1}{2}\Pr[\mathcal{A}^{hy} \text{ outputs } 0 | \bar{k} = k, m = m_0] \\ + \frac{1}{2}\Pr[\mathcal{A}^{hy} \text{ outputs } 1 | \bar{k} = k, m = m_1]\end{aligned}$$

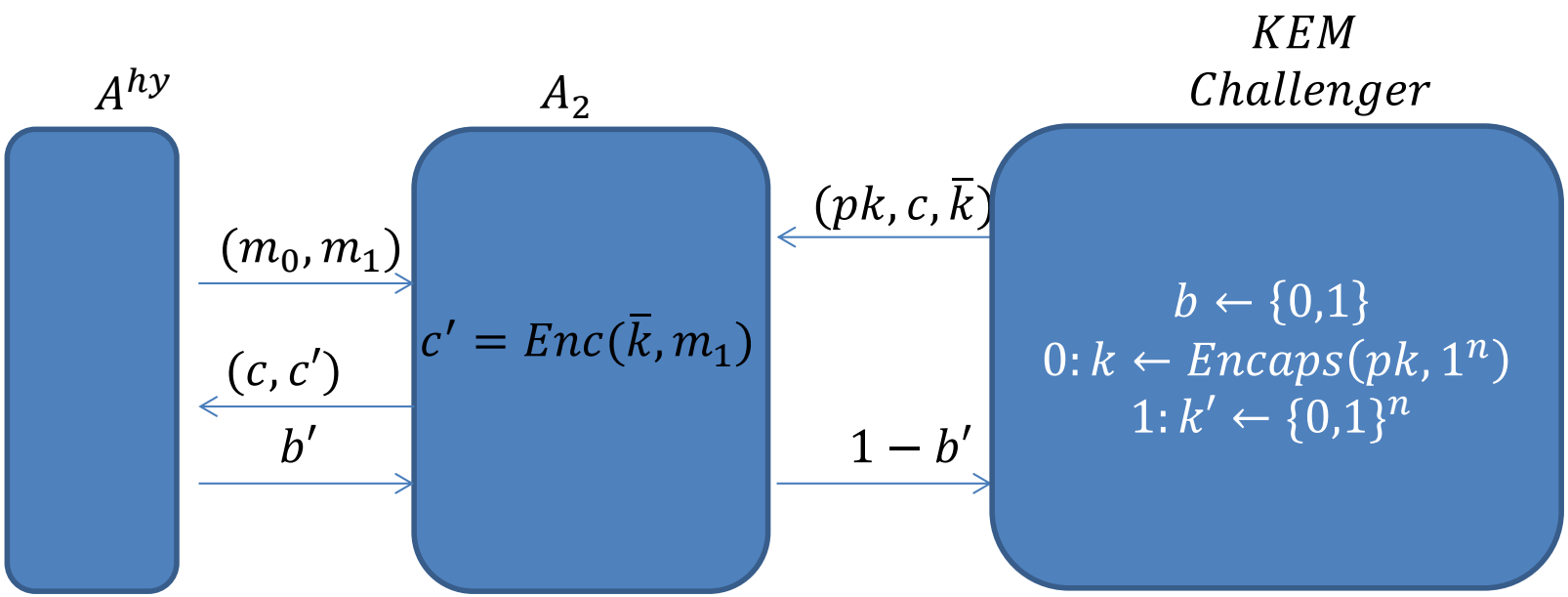
Security of the Hybrid Encryption Scheme



$$\mathbf{Pr}[A_1'\boldsymbol{s\ output} = \mathbf{0}|\boldsymbol{b} = \mathbf{0}] = \mathbf{Pr}[A^{hy'}\boldsymbol{s\ output} = \mathbf{0}|\bar{\boldsymbol{k}} = \boldsymbol{k}, \boldsymbol{m} = \boldsymbol{m_0}]$$

$$\mathbf{Pr}[A_1'\boldsymbol{s\ output} = \mathbf{1}|\boldsymbol{b} = \mathbf{1}] = \mathbf{Pr}[A^{hy'}\boldsymbol{s\ output} = \mathbf{1}|\bar{\boldsymbol{k}} = \boldsymbol{k'}, \boldsymbol{m} = \boldsymbol{m_0}]$$

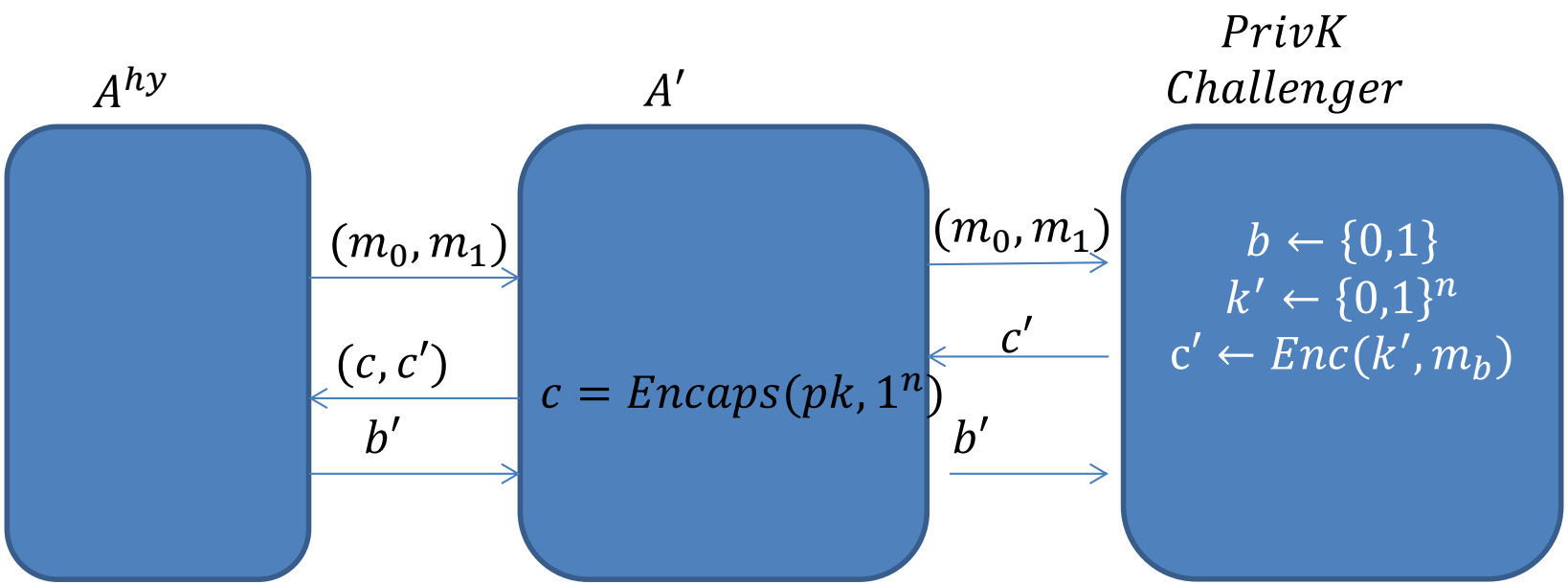
Security of the Hybrid Encryption Scheme



$$\mathbf{Pr}[\boldsymbol{A_2's\ output} = \mathbf{0} | \boldsymbol{b} = \mathbf{0}] = \mathbf{Pr}[\boldsymbol{A^{hy'}s\ output} = \mathbf{1} | \bar{\boldsymbol{k}} = \boldsymbol{k}, \boldsymbol{m} = \boldsymbol{m_1}]$$

$$\mathbf{Pr}[\boldsymbol{A_2's\ output} = \mathbf{1} | \boldsymbol{b} = \mathbf{1}] = \mathbf{Pr}[\boldsymbol{A^{hy'}s\ output} = \mathbf{0} | \bar{\boldsymbol{k}} = \boldsymbol{k'}, \boldsymbol{m} = \boldsymbol{m_1}]$$

Security of the Hybrid Encryption Scheme



$$\mathbf{Pr}[A' \textbf{ output} = \mathbf{0} | b = 0] = \mathbf{Pr}[A^{hy'} \textbf{s output} = \mathbf{0} | \bar{k} = k', m = m_0]$$

$$\mathbf{Pr}[A' \textbf{ output} = \mathbf{1} | b = 1] = \mathbf{Pr}[A^{hy'} \textbf{s output} = \mathbf{1} | \bar{k} = k', m = m_1]$$

Security of the Hyprid Encryption Scheme

Theorem

The hybrid encryption scheme is a CCA-secure public-key encryption scheme if KEM is CCA secure and the private-key encryption scheme is CCA-secure.

Outline

RSA Encryption Scheme

- Designed by Rivest-Shamir-Adleman in 1977
- It is widely in use today. There is also the RSA digital signature scheme.
- Security of both *relies* on the fact that integer factorization is a hard computational problem.

Pseudorandom Permutations from One Way Functions

- Informally speaking, *one-way functions* are easy to compute, hard to invert!
- We don't know how to prove that one-way functions exist!
- Assuming the hardness of some problems, we can build one-way functions.

Corollary

Let $n > 1$, *and for* $e > 0$ *define* $f_e : \mathbb{Z}_n^* \to \mathbb{Z}_n^*$ *by* $f_e(x) = x^e \mod n$. *If* $GCD(e, \phi(n)) = 1$, *then* f_e *is a permutation. The inverse of* f_e *is* f_d *where* $d = e^{-1} \mod \phi(n)$

Chinese Remainder Theorem

- If $n = \prod_{i=1}^{N} p_i^{e_i}$ then the map

$$f : \mathbb{Z}_n \to \prod_{i=1}^{N} \mathbb{Z}_{p_i^{e_i}} : x \to (x \bmod p_1^{e_1}, \ldots, x \bmod p_N^{e_N})$$

  is a ring isomorphism
- In other words given all residue values, there exists a unique value that corresponds to them modulo n

Euler's theorem

- Let $n = \prod_{i=1}^{N} p_i^{e_i}$ where the p_i are distinct primes
- The Euler totient function $\phi(n)$ is the number of positive integers less than or equal to n that are relatively prime to n, more formally,

$$\phi(n) = \prod_{i=1}^{N} (p_i - 1) p_i^{e_i - 1}$$

- Then for all $x \in \mathbb{Z}_n^*$, we have

$$x^{\phi(n)} = 1 \bmod n$$

- If $n = p$ a prime, then $\phi(n) = p - 1$ and we recover Fermat's little theorem $x^{p-1} = 1 \bmod p$
- If $n = pq$ like in RSA, then $\phi(n) = (p-1)(q-1)$

Plain RSA encryption algorithm

- Let p, q two distinct odd primes, and let $n = pq$
- Compute $\phi(n) = (p-1)(q-1)$, and choose $e > 1$ s.t. $\gcd(e, \phi(n)) = 1$
- **Public key** is (n, e) and **private key** is (p, q)
- Given private key, can also compute $d := e^{-1} \bmod \phi(n)$
- Encryption of $m \in \mathbb{Z}_n^*$: $c = m^e \bmod n$
- Decryption of $c \in \mathbb{Z}_n^*$: $m' = c^d \bmod n$
- Correctness follows from

$$m' = (m^e)^d = m^{ed \bmod \phi(n)} = m \bmod n$$

by Euler's theorem

RSA security

- Solving the factorization problem is sufficient and necessary to reconstruct the private key
- Solving the factorization problem *might not be necessary* for other goals, such as decrypting without the private key
- In fact, “Plain RSA” is insecure!
  - What if m is not chosen uniformly from $\mathbb{Z}_n^*$?
  - Plain RSA is deterministic!
  - Therefore, it is not CPA-secure!

Further Reading (1)

Mihir Bellare, Alexandra Boldyreva, and Silvio Micali.
Public-key encryption in a multi-user setting: Security proofs and improvements.
In Bart Preneel, editor, *Advances in Cryptology — EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 259–274. Springer Berlin Heidelberg, 2000.

Dan Boneh.
Simplified OAEP for the RSA and Rabin Functions.
In Joe Kilian, editor, *Advances in Cryptology — CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 275–291. Springer Berlin Heidelberg, 2001.

Further Reading (2)


Ronald Cramer and Victor Shoup.
Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack.
SIAM Journal on Computing, 33(1):167–226, 2003.


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

Further Reading (3)

Itai Dinur, Orr Dunkelman, Nathan Keller, and Adi Shamir.
New attacks on feistel structures with improved memory complexities.
In *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, pages 433–454, 2015.

Naofumi Homma, Atsushi Miyamoto, Takafumi Aoki, Akashi Satoh, and Adi Shamir.
Collision-based power analysis of modular exponentiation using chosen-message pairs.
In *Cryptographic Hardware and Embedded Systems - CHES 2008, 10th International Workshop, Washington, D.C., USA, August 10-13, 2008. Proceedings*, pages 15–29, 2008.