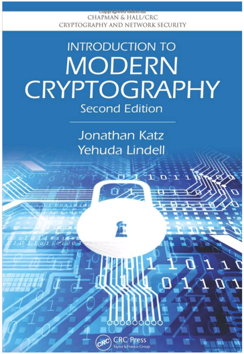
Public Key Cryptography

Ali El Kaafarani

[1]Mathematical Institute
[2] PQShield Ltd.

Outline

Course main reference



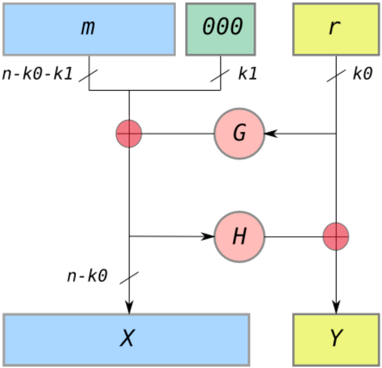
Padded RSA: RSA#1 v1.5

- Idea: To encrypt a message m, first map it to an element $\tilde{m} \in \mathbb{Z}_n^*$.
- The sender can choose a uniform bit-string $r \in \{0, 1\}^\ell$, and sets $\tilde{m} = r||m$ (it is a reversible operation).
- The security of the padded scheme depends on the length of ℓ. The cost of a brute-force attack is $\mathcal{O}\left(2^\ell\right)$
- For instance, $\ell(n) = \mathcal{O}(\log n)$ is a bad choice, the scheme is not secure in this case.
- This scheme is provably secure based on the RSA problem in one case: ℓ is very large, and m is just a single bit!
- For other cases, no security proofs based on RSA problem, BUT no known attacks are known either!

RSA-OAEP

- It is a construction that is based on RSA problem and CCA-secure using *optimal asymmetric encryption padding* OAEP.
- Already standardized as a part of RSA PKCS#1 since version 2.0
- It uses three integer-valued functions $\ell(n), k_0(n), k_1(n)$ with $k_0(n), k_1(n) = \Theta(n)$. There is also a condition on $\ell(n) + k_0(n) + k_1(n)$, it has to be smaller than the minimum bit-length of RSA moduli.
- We need two hash functions H and G that are modelled as *Random Oracles*
- OAEP is therefore a two-round Feistel network. G and H are the round functions.

RSA-OAEP



Source: Wikipedia

RSA-OAEP

Fix n and let $\ell = \ell(n), k_0 = k_0(n), k_1 = k_1(n)$. Given $H : \{0,1\}^{\ell+k_1} \to \{0,1\}^{k_0}$ and $G : \{0,1\}^{k_0} \to \{0,1\}^{\ell+k_1}$.

How to pad a message $m \in \{0,1\}^{\ell}$?

- Set $m' \leftarrow m||0^{k_1}$
- Choose a random $r \in \{0,1\}^{k_0}$
- Compute $s \leftarrow m' \oplus G(r) \in \{0,1\}^{\ell+k_1}$
- Compute $t \leftarrow r \oplus H(s) \in \{0,1\}^{k_0}$
- Finally, set $\tilde{m} \leftarrow s||t$.

RSA-OAEP

How does it work?

- KeyGen(n) : output the public key (n, e) and private key (p, q).
- Enc(m, N, e) : pad m to get $\tilde{m}$. The ciphertext will be $c \leftarrow \tilde{m}^e \mod n$.
- Dec(c, n, d) : compute $\tilde{m} \leftarrow c^d \mod n$. If $|\tilde{m}| > \ell + k_0 + k_1$, output $\perp$, otherwise;
  - parse $\tilde{m}$ as $s||t$, $s \in \{0,1\}^{\ell+k_1}, t \in \{0,1\}^{k_0}$
  - compute $r \leftarrow H(s) \oplus t$
  - compute $m' \leftarrow G(r) \oplus s$ if the least-significant k_1 bits of m' are not all 0, output $\perp$
  - otherwise, output the ℓ **most-significant bits of** $\tilde{m}$.

A CCA secure KEM in the ROM

The KEM scheme consists of the following algorithms:

- KeyGen(1^n): it generates the RSA modulus (N, e, d), where PK $= (N, e)$ and SK $= (N, d)$. it also generates a hash function $H : \mathbb{Z}_N^* \to \{0, 1\}^n$.
- Encaps(PK$, 1^n)$: it picks a random $r \in \mathbb{Z}_N^*$ and outputs $c \leftarrow r^e \mod N$ and the key $k \leftarrow H(r)$.
- Decaps(SK$, c \in \mathbb{Z}_N^*)$: it first computes $r \leftarrow c^d \mod N$ and then outputs $k \leftarrow H(r)$.

This is a part of ISO/IEC18033-2 standard for public-key encryption.

Security of RSA-OAEP

- It is CCA-secure assuming that G and H are modelled as random oracles.
- There were some attacks on PKCS# v2.0 in 2001 by James Manger that exploits its implementation- it is a side channel attack!
- The receiver receives the error message $\perp$ in two different cases!
- The time to return the message errors was not identical.
- The attacker can recover a message m using ONLY $|N|$ queries.
- Lesson: side channels attacks are nasty! Implementations should take into consideration every possibility of information leakage!

RSA weak key generator attack

- Suppose Alice uses private key (p, q_a) and Bob uses private key (p, q_b). Is it safe?

RSA weak key generator attack

- Suppose Alice uses private key (p, q_a) and Bob uses private key (p, q_b). Is it safe?
- Everybody sees $n_a := pq_a$ and $n_b := pq_b$

RSA weak key generator attack

- Suppose Alice uses private key (p, q_a) and Bob uses private key (p, q_b). Is it safe?
- Everybody sees $n_a := pq_a$ and $n_b := pq_b$
- Alice can compute $q_b = n_b/p$
- Bob can compute $q_a = n_a/p$

RSA weak key generator attack

- Suppose Alice uses private key (p, q_a) and Bob uses private key (p, q_b). Is it safe?
- Everybody sees $n_a := pq_a$ and $n_b := pq_b$
- Alice can compute $q_b = n_b/p$
- Bob can compute $q_a = n_a/p$
- **Anyone** can compute $\gcd(n_a, n_b) = p$ and then q_a and q_b
- Attack demonstrated in practice

  Lenstra et al. *Ron was wrong, Whit is right*
  Show that 2/1000 RSA keys are insecure

Outline

The Discrete Logarithm Problem (Dlog)

- Let p be a prime and let $K := \mathbb{F}_p = \mathbb{Z}/p\mathbb{Z}$
- Exponentiation in K in $\mathcal{O}(n) = \mathcal{O}(\log p)$ multiplications
- What about the inverse operation?
- **Discrete logarithm problem:**
  Given g and $h = g^k \bmod p$, compute k
- Believed to be very hard: subexponential complexity $L_p(1/3, c)$
- More generally: given G, $g \in G$ and $h = g^k$, compute k
- Can be harder or easier depending on the group

Diffie-Hellman Key Exchange Algorithm

- Designed by Diffie and Hellman in 1976.
- Public elements: G cyclic, $g \in G$ a generator
- Alice chooses random a and sends g^a to Bob
- Bob chooses random b and sends g^b to Alice
- Alice computes $(g^b)^a = g^{ab}$
- Bob computes $(g^a)^b = g^{ab}$

Variants of Diffie-Hellman Problem

- Computational Diffie-Hellman (CDH): Given $g, g^a, g^b \in G$, compute g^{ab}.
- Decisional Diffie-Hellman (DDH): Given $g, h, g^a, g^b \in G$, decide if $h = g^{ab}$.
- There is a huge list of members in the DH family of problems!

Diffie-Hellman security

- Solving discrete logarithm problem is sufficient to break Diffie-Hellman key exchange
- Solving discrete logarithm problem *might not* be necessary to break Diffie-Hellman key exchange
- For authentication, we use certificate.

Outline

ElGamal Encryption Scheme

Main idea:

- Given a finite group $\mathbb{G}$, let m be an arbitrary element of $\mathbb{G}$.
- Lemma: if we multiply m by an uniform group element of $\mathbb{G}$, say k, the result $k \cdot m$ is a uniform group element.
- Proof: Let g be an arbitrary element of G,

$$\Pr[k \cdot m = g] = \Pr[k = g \cdot m^{-1}].$$

  And because k is uniform

$$\Pr[k = g \cdot m^{-1}] = 1/|\mathbb{G}|.$$

ElGamal Scheme- Construction

We define ElGamal public key encryption scheme as follows:

- KeyGen(n) : first, it outputs a description a cyclic group $\mathbb{G}$ with order q, where $|q| = n$ and a generator g, i.e $(\mathbb{G}, q, g)$. Then, it picks a uniform $x \in \mathbb{Z}_q$ to compute $h \leftarrow g^x$. the public key is PK $= (\mathbb{G}, g, q, h)$ and the private/secret key is SK $= x$. The messages are elements of $\mathbb{G}$.
- Enc(PK$, m \in \mathbb{G})$: it chooses a uniform $y \in \mathbb{Z}_q$, and output the following ciphertext

$$c = (c_1, c_2) \leftarrow (g^y, h^y \cdot m).$$

- Dec(SK$, c)$: it outputs

$$m' = c_2 / c_1^x$$

- Check its correctness!

ElGamal scheme- Example

Example

[Katz-Lindell book] Let $q = 83$ and $p = 2q + 1 = 167$. Let $\mathbb{G}$ denote the group of quadratic residues mod p. As both p and q are primes, then $\mathbb{G}$ is a subgroup of $\mathbb{Z}_p^*$ with order q. Note that $|\mathbb{G}|$ is prime, so any element $1 \neq g \in \mathbb{G}$ is a generator. Take $g = 2^2 = 4 \mod 167$, pick $x = 37 \in \mathbb{Z}_{83}$, compute $h = g^x = 4^{37} \mod 167 = 76$
The public becomes PK $= (p, q, g, h) = (167, 83, 4, 76)$

- Enc(PK, $m = 65 \in \mathbb{G}$): [a] it picks $y = 71$ and compute the ciphertext,

$$c = (c_1, c_2) = (4^{71}, 76^{71} \cdot 65) = (132, 44) \mod 167$$

[a] 65 is indeed in $\mathbb{G}$ as $65 = 30^2 \mod 167$

ElGamal Scheme-Example

Example

- Dec(SK, c):

$$\begin{aligned} m =& c_2/c_1^x \\ =& 44/132^{37} \mod 167 \\ =& 44/124 \mod 167 \\ =& 44 \cdot 124^{-1} \mod 167 \\ =& 44 \cdot 66 \mod 167 \\ =& 65 \end{aligned}$$

Security of ElGamal Scheme

Theorem

If the DDH problem is hard, then the ElGamal encryption scheme is CPA-secure.

Sketch Proof.

Idea: we consider a PPT algorithm D that wants to solve DDH, and PPT algorithm A *(the adversary) who is attacking ELGamal scheme S. the algorithm D first receives an instance of the DDH problem, i.e $(\mathbb{G}, q, g, h_1 = g^x, h_2 = g^y, h_3)$,and his challenge is to figure out whether or not h_3 is equal to g^{xy}.*

Security of ElGamal Scheme

Sketch Proof.

Algorithm $\mathcal{D}$ will simulate the ElGamal scheme to $\mathcal{A}$ as follows:

- *On input $(\mathbb{G}, q, g, h_1, h_2, h_3)$, it sets* $\mathsf{PK} = (\mathbb{G}, q, g, h_1)$.
- *On input (m_0, m_1) received from $\mathcal{A}$, it picks $b \in \{0, 1\}$, and sets $c_1 = h_2$ and $c_2 = h_3 \cdot m_b$ and sends them over to $\mathcal{A}$*
- *It receives the bit b' from $\mathcal{A}$, it then outputs 1 if $b' = b$ and 0 otherwise.*

Now, let S' be a modified version of ElGamal, works as follows:

- *Same key generation algorithm*
- *Encryption algorithm: it chooses a uniform $y, z \in \mathbb{Z}_q$, and output the following ciphertext $(g^y, g^z \cdot m)$. Note that the decryption algorithm doesn't work here, but we don't actually need it in the experiment.*

Security of ElGamal Scheme

Sketch Proof.

For the modified encryption scheme, since c_2 is a uniformly distributed group element, we have

$$\Pr[\mathsf{PubK}^{CPA}_{\mathcal{A},S'}(n) = 1] = 1/2$$

And if DDH holds, then

$$|\Pr[D(\mathbb{G}, q, g, g^x, g^y, g^z) = 1] - \Pr[D(\mathbb{G}, q, g, g^x, g^y, g^{xy}) = 1]| < \mathsf{negl}(n) \quad (1)$$

Case 1–random tuple: *We can easily see that the View $\mathcal{A}$ when run as a subroutine by D is distributed identically to its view in experiment* $\mathsf{PubK}^{cpa}_{\mathcal{A},S'}$*. Therefore*

$$\Pr[D(\mathbb{G}, q, g, g^x, g^y, g^z) = 1] = \Pr[\mathsf{PubK}^{CPA}_{\mathcal{A},S'}(n) = 1] = 1/2 \quad (2)$$

Security of ElGamal Scheme

Sketch Proof.

Case 2– DH tuple: *We can also see that the View $\mathcal{A}$ when run as a subroutine by D is distributed identically to its view in experiment* $\mathsf{PubK}^{cpa}_{\mathcal{A},S}$*. Therefore*

$$\Pr[\mathsf{PubK}^{CPA}_{\mathsf{A},S}(n) = 1] = \Pr[D(\mathbb{G}, q, g, g^x, g^y, g^{xy}) = 1] \tag{3}$$

Equations (1), (2) and (3) give us

$$\Pr[\mathsf{PubK}^{CPA}_{\mathsf{A},S}(n)] < 1/2 + \mathsf{negl}(n)$$

Security of ElGamal Scheme

Sketch Proof.

Case 2– DH tuple: *We can also see that the View $\mathcal{A}$ when run as a subroutine by D is distributed identically to its view in experiment* $\mathsf{PubK}^{cpa}_{\mathcal{A},S}$*. Therefore*

$$\Pr[\mathsf{PubK}^{CPA}_{\mathsf{A},S}(n) = 1] = \Pr[D(\mathbb{G}, q, g, g^x, g^y, g^{xy}) = 1] \tag{3}$$

Equations (1), (2) and (3) give us

$$\Pr[\mathsf{PubK}^{CPA}_{\mathsf{A},S}(n)] < 1/2 + \mathsf{negl}(n)$$

Is ElGamal scheme CCA-secure?

Security of ElGamal Scheme

Sketch Proof.

Case 2– DH tuple: *We can also see that the View $\mathcal{A}$ when run as a subroutine by D is distributed identically to its view in experiment* $\mathsf{PubK}^{cpa}_{\mathcal{A},S}$*. Therefore*

$$\Pr[\mathsf{PubK}^{CPA}_{\mathsf{A},S}(n) = 1] = \Pr[D(\mathbb{G}, q, g, g^x, g^y, g^{xy}) = 1] \tag{3}$$

Equations (1), (2) and (3) give us

$$\Pr[\mathsf{PubK}^{CPA}_{\mathsf{A},S}(n)] < 1/2 + \mathsf{negl}(n)$$

Is ElGamal scheme CCA-secure?Why?

A CPA-secure KEM Scheme based on DDH

The scheme consists of the following algorithms:

- KeyGen(1^n): it generates $(\mathbb{G}, q, g)$. It then chooses $x \in \mathbb{Z}_q$ and computes $h = g^x$. It also specifies a hash function $H : \mathbb{G} \to \{0,1\}^{\ell(n)}$ The public key PK $= (\mathbb{G}, q, g, h, H)$ and the private key is x.
- Encaps(PK): it chooses a uniform $y \in \mathbb{Z}_q$ and outputs the ciphertext $c \leftarrow g^y$ and the key $H(h^y)$.
- Decaps(SK, c): it outputs $H(c^x)$.

If H is modelled as a random oracle model, then the scheme is CPA-secure based on (the weaker assumption) CDH

Outline

Cramer-Shoup cryptosystem

- The first public key encryption scheme that is CCA-secure without random oracles.
- It is based on ElGamal.
- Its CCA-security relies on the hardness of DDH.

Cramer-Shoup Cryptosystem

- KeyGen(n) : first, it outputs a description a cyclic group $\mathbb{G}$ with prime order q, s.t. $\|q\| = n$ and a couple of generators g_1, g_2, i.e $(\mathbb{G}, q, g_1, g_2)$. Then, it picks a uniform $x_1, x_2, y_1, y_2, z_1, z_2 \in \mathbb{Z}_q$, it computes
  - $c \leftarrow g_1^{x_1} g_2^{x_2}$
  - $d \leftarrow g_1^{y_1} g_2^{y_2}$
  - $h \leftarrow g_1^{z_1} g_2^{z_2}$

  The public key is PK $= (\mathbb{G}, q, g_1, g_2, c, d, h, H)$ where $H()$ is a collision-resistant hash function. The private/secret key is SK $= (x_1, x_2, y_1, y_2, z_1, z_2)$. The messages are elements of $\mathbb{G}$.

Cramer-Shoup Cryptosystem

- Enc(PK, $m \in \mathbb{G}$) : it chooses a uniform $k \in \mathbb{Z}_q$, and outputs the following ciphertext:
  - $u_1 = g_1^k, u_2 = g_2^k$
  - $e = h^k m$
  - $\alpha = H(u_1, u_2, e)$
  - $v = c^k d^{k\alpha}$

  The ciphertext is $CT = (u_1, u_2, e, v)$

Cramer-Shoup Cryptosystem

- Dec(CT, SK) :
  - It computes $\alpha = H(u_1, u_2, e)$,
  - If $u_1^{x_1} u_2^{x_2} (u_1^{y_1} u_2^{y_2})^{\alpha} \neq v$, output $\perp$
  - It outputs $m' = e/(u_1^{z_1} u_2^{z_2})$

Correctness:

$$m' = e/(u_1^{z_1} u_2^{z_2}) = h^k m / g_1^{kz_1} g_2^{kz_2} = h^k m / h^k = m$$

Cramer-Shoup: Security Proof

Let $\mathcal{A}$ be the adversary attacking the Cramer-Shoup scheme and $\mathcal{D}$ the distinguisher that wants to distinguish a DH tuple from a random tuple.

Proof.

Distinguisher $\mathcal{D}(g_1, g_2, g_3, g_4)$

$x_1, x_2, y_1, y_2, z_1, z_2 \leftarrow \mathbb{Z}_q$.
PK $= (g_1, g_2, c := g_1^{x_1} g_2^{x_2}, d := g_1^{y_1} g_2^{y_2}, h := g_1^{z_1} g_2^{z_2}, H)$.
$(m_0, m_1) \leftarrow \mathcal{A}(\mathsf{PK}, \mathsf{Dec}(\mathsf{SK}, \cdot))$.
$b \leftarrow \{0, 1\}$.
$CT^* = (g_3, g_4, g_3^{z_1} g_4^{z_2} m_b, g_3^{x_1+\alpha y_1} g_4^{x_2+\alpha y_2})$.
$b' \leftarrow \mathcal{A}(\mathsf{PK}, CT^*, \mathsf{Dec}(\mathsf{SK}, \cdot)^{CT^*})$,
Output 1 iff $b' = b$ $\square$

Cramer-Shoup: Security Proof

Proof.

Claim 1:

$$\Big|\Pr[D = 1|DH] - \Pr[D = 1|\text{Random}]\Big| = \text{negl}(n)$$

[It follows from the DDH assumption]

Claim 2:

$$\Pr[D \text{ outputs } 1|\text{DH}] = \Pr[b' = b|\mathcal{A} \text{ attacks } S \text{ directly}]$$

Claim 3:

$$\Big|\Pr[D = 1|\text{Random}] - \frac{1}{2}\Big| = \text{negl}(n)$$

□

Cramer-Shoup: Security Proof

Proof.

When $\mathcal{D}$ gets a DH tuple, then there exist γ, r s.t.:
$(g_1, g_2 = g_1^{\gamma}, g_3 = g_1^{r}, g_4 = g_2^{r})$
It is easy to verify that the distribution of PK and CT are exactly the same of those obtained from a real world Cramer-Shoup challenger (and not from the distinguisher who is simulating the game). Therefore,

$$\Pr[D \text{ outputs } 1 | \text{DH tuple}] = \Pr[b' = b | \mathcal{A} \text{ attacks } S \text{ directly}]$$

i.e.

$$\Pr[D \text{ outputs } 1 | \text{DH tuple}] = \Pr[\mathsf{PubK}^{cca}_{\mathcal{A},CS}(n) = 1]$$

$\square$

Cramer-Shoup: Security Proof

Proof.

When $\mathcal{D}$ gets a random tuple, it will look like $(g_1, g_2 = g_1^{\gamma}, g_3 = g_1^{r}, g_4 = g_2^{r'})$, where $\gamma \neq 0$ and $r \neq r'$.

Getting information about z_1, z_2:

- From the PK, $\mathcal{A}$ learns

$$\log_{g_1} h = z_1 + \gamma z_2. \tag{4}$$

- From the decryption oracle on $CT = (u_1, u_2, e, v)$, we distinguish between two cases, valid and invalid ciphertexts. We will prove that he learns nothing from valid ciphertexts and that the probability that it accepts invalid ciphertexts is negligible. CT is invalid if $\log_{g_1} u_1 \neq \log_{g_2} u_2$ and Dec(SK, $\cdot$) doesn't return $\perp$, it is valid otherwise.

Cramer-Shoup: Security Proof

Proof.

no extra information from valid ciphertexts, why?

When Dec() returns $\perp$, it means that v is not in the right format, but z_1, z_2 are not involved in this check, so no information about them in this case.
On the other hand, if

$$\log_{g_1} u_1 = \log_{g_2} u_2 = r''$$

then what $\mathcal{A}$ can learn from m is

$$\log_{g_1} m = \log_{g_1} e - r'' z_1 - r'' \gamma z_2 \tag{5}$$

But equation (5) is linearly dependent on equation (4), so no extra information about z_1, z_2 from this case. $\square$

Cramer-Shoup: Security Proof

During the course of the experiment, $\mathcal{A}$ learns the following about x_1, x_2, y_1, y_2:

Proof.

From the public key, $\mathcal{A}$ learns the following:

$$\log_{g_1} c = x_1 + x_2\gamma \tag{6}$$

$$\log_{g_1} d = y_1 + y_2\gamma \tag{7}$$

From the challenge ciphertext, $\mathcal{A}$ learns:

$$\log_{g_1} v^* = (x_1 + \alpha y_1)r + (x_2 + \alpha y_2)\gamma r' \tag{8}$$

□

Cramer-Shoup: Security Proof

Proof.

Now the idea is to prove that the probability that $\mathcal{A}$ submits the previous type of "bad" decryption queries is negligible. Let $CT^* = (u_1^*, u_2^*, e^*, v^*)$ be the challenge ciphertext, we have three possible types of "bad" decryption queries:

- $(u_1, u_2, e) = (u_1^*, u_2^*, e^*)$ with $v \neq v^*$. Since we will have same hash values but with $v \neq v^*$, the decryption oracle will reject it.
- $(u_1, u_2, e) \neq (u_1^*, u_2^*, e^*)$ and $\alpha = \alpha'$. It means that we found a collision in H, but H is collision-resistant, so this happens only with negligible probability.

□

Cramer-Shoup: Security Proof

Proof.

- $(u_1, u_2, e) \neq (u_1^*, u_2^*, e^*)$ and $\alpha \neq \alpha'$. The decryption oracle will accept the query only if

$$\log_{g_1} v = (x_1 + \alpha' y_1)\tilde{r} + (x_2 + \alpha' y_2)\gamma\tilde{r}' \tag{9}$$

  where $\tilde{r} \neq \tilde{r}'$, is linearly dependent with (6),(7),(8).

BUT, we can show that in this case, the equations (6),(7),(8) and (9) are linearly independent because

$$\det \begin{pmatrix} 1 & \gamma & 0 & 0 \\ 0 & 0 & 1 & \gamma \\ r & r'\gamma & r\alpha & r'\alpha\gamma \\ \tilde{r} & \tilde{r}'\gamma & \tilde{r}\alpha' & \tilde{r}'\alpha'\gamma \end{pmatrix} = (\gamma^2)(r' - r)(\tilde{r}' - \tilde{r})(\alpha - \alpha') \neq 0$$

□

Cramer-Shoup: Security Proof

Proof.

In the third case, the decryption query is rejected except with probability $1/q$, which is the probability to have v in the right format (from $\mathcal{A}$'s point of view, v is uniformly distributed in $\mathbb{G}$), this v has to use the same values for x_1, x_2, y_1, y_2 that are used in (6),(7),(8) (remember that these values are unknown to $\mathcal{A}$). If the adversary makes η queries, then the probability that one of these queries is not rejected is at most $\dfrac{\eta}{q - \eta + 1}$ which is negligible as q is exponential in the security parameter whereas the number of queries η is polynomial in it.

We deduce that the hidden bit b is independent from $\mathcal{A}$'s view except when either a collision is found in H or the decryption oracle accepts an invalid ciphertext. Claim 3 follows.

□

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.