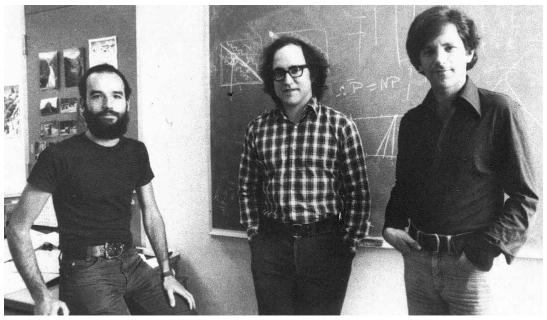
Crypto 9

Note: this material is not intended to replace the live lecture for students.

Contents



9.1 Key Exchange

Key-Exchange Protocol

is a protocol Π i.e. a set of instructions for the parties say Alice and Bob, that starting from a security parameter n allows them to compute keys k_A and k_B.
The correctness requirement is that

$$\mathsf{k}_A = \mathsf{k}_B$$

so we can speak simply of *the* key $\mathsf{k} = \mathsf{k}_A = \mathsf{k}_B$ as the exchanged key.

9.1.1 Diffie-Hellman (DH)

The best known cryptographic problem is that of privacy: preventing the unauthorized extraction of information from communications over an insecure channel. In order to use cryptography to insure privacy, however, it is currently necessary for the communicating parties **to share a key** which is known to no one else. This is done by sending the key in advance over some secure channel such as private courier or registered mail. A private conversation between two people with no prior acquaintance-is a common occurrence in business, however, and **it is unrealistic to expect initial business contacts to be postponed long enough for keys to be transmitted by some physical means.** *The cost and delay imposed by this key distribution problem is a major barrier to the transfer of business communications to large teleprocessing networks.* ☺

[DH76, Introduction, page 644]

This fundamental key agreement technique is implemented in many open and commercial cryptographic protocols like Secure Shell (SSH), Transport Layer Security (TLS), and Internet Protocol Security (IPSec).

9.1.1 Diffie-Hellman (DH)

Alice and **Bob** agree to use a prime number p and an element g of $\mathrm{GF}^*(p)$.

$$\begin{array}{rcl} \textbf{Alice} & & \textbf{Bob} \\ \text{Pick RND } A \in \{1, \cdots, p-1\} & & \\ \text{Compute } g^{A} \pmod p & & \\ & \xrightarrow{\quad g^{A} \quad} & \\ & & \text{Pick RND } B \in \{1, \cdots, p-1\} \\ & & \text{Compute } g^{B} \pmod p \\ & \xleftarrow{\quad g^{B} \quad} & \\ \mathsf{k}_A = (g^{B})^{A} & & \mathsf{k}_B = (g^{A})^{B} \end{array}$$

So $\mathsf{k} = \mathsf{k}_A = \mathsf{k}_B$ is the session key between **Alice** and **Bob**.

A and B are the private keys whilst g^A, g^B are the public keys.

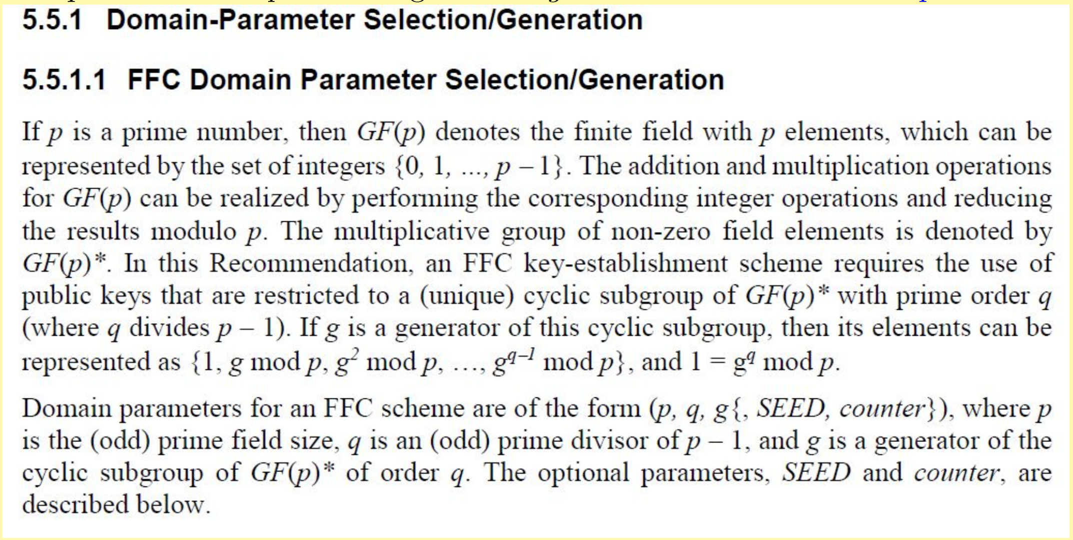
Exercise 9.1.2

Check that

$$\mathsf{k}_A = (g^{B})^{A} = \mathsf{k}_B = (g^{A})^{B} \pmod p$$

NOTE 9.1.3

The prime number p and the generator g are the so called domain parameters:



5.5.1 Domain-Parameter Selection/Generation

5.5.1.1 FFC Domain Parameter Selection/Generation

If p is a prime number, then $GF(p)$ denotes the finite field with p elements, which can be represented by the set of integers $\{0, 1, ..., p-1\}$. The addition and multiplication operations for $GF(p)$ can be realized by performing the corresponding integer operations and reducing the results modulo p. The multiplicative group of non-zero field elements is denoted by $GF(p)^*$. In this Recommendation, an FFC key-establishment scheme requires the use of public keys that are restricted to a (unique) cyclic subgroup of $GF(p)^*$ with prime order q (where q divides $p-1$). If g is a generator of this cyclic subgroup, then its elements can be represented as $\{1, g \bmod p, g^2 \bmod p, \ldots, g^{q-1} \bmod p\}$, and $1 = g^q \bmod p$.

Domain parameters for an FFC scheme are of the form (p, q, g\{, *SEED*, *counter*\}), where p is the (odd) prime field size, q is an (odd) prime divisor of $p-1$, and g is a generator of the cyclic subgroup of $GF(p)^*$ of order q. The optional parameters, *SEED* and *counter*, are described below.

In applications p should have at least 2048 bits to avoid attacks as index calculus. More info here: `https://www.keylength.com/en/compare/`.

9.1.4 Man in the Middle (MITM)

Alice | **Oscar** | **Bob**

Pick RND A | | Pick RND B

$\xrightarrow{g^A}$ | | $\xleftarrow{g^B}$

Pick OA, OB

$\xleftarrow{g^{OA}}$ | | $\xrightarrow{g^{OB}}$

$\mathsf{k}_A = (g^{OA})^A$ | | $\mathsf{k}_B = (g^{OB})^B$

$\mathsf{k}_A = (g^A)^{OA}$

$\mathsf{k}_B = (g^B)^{OB}$

and from there Oscar is going read all messages between **Alice** and **Bob**

NOTE 9.1.5

The DH Problem (DHP) is to compute $g^{A \cdot B}$ from g^A and g^B. It is not know if DHP is equivalent to the discrete logarithm problem.

Attacks

Active: To avoid MITM should be used certificates from a trusted third party.

Passive: Discrete Logarithm one-wayness.

Exercise 9.1.6

Consider DH key agreement with modulus p of 1024 bit and g a generator of a subgroup of order $\approx 2^{160}$.

i) What is the maximum value that the private keys should have?

ii) How long does the computation of the session key take on average if one modular multiplication takes 700 μs, and one modular squaring 400 μs? Assume that the public keys have already been computed.

9.2 Public Key Encryption

Public Key Cryptosystem (PKC)

Such cryptosystem consists of three algorithms

$$(\mathsf{Gen}\,,\ \mathsf{Enc}\,,\ \mathsf{Dec})$$

Gen generate a pair (sk,pk) of secret key sk and public key pk.

1) The algorithms $\mathsf{Gen}, \mathsf{Enc}$ e Dec must be computationally feasible,

2) The secret key sk should be computationally infeasible to compute from the public key pk.

9.2.1 Textbook Elgamal

The Elgamal encryption as proposed in 1985 by Taher Elgamal [Elgamal85] is widely used. For example, Elgamal is part of the free GNU Privacy Guard (GnuPG), OpenSSL, Pretty Good Privacy (PGP).

9.2.1 Elgamal

- Parameter domains: g and p as in DH key agreement.
- $\mathsf{Gen}(\lambda)$: pick $A \in \{1, \cdots, p-1\}$ and compute $h = g^A$. Set $\mathsf{pk} = h$ and $\mathsf{sk} = A$.
- $\mathsf{Enc}_{\mathsf{pk}}(m)$ with $m \in \mathrm{GF}(p)$: pick RND $B \in \{1, \cdots, p-1\}$ and compute $C = (g^B, m \cdot h^B)$.
- $\mathsf{Dec}_{\mathsf{sk}}(C)$ with $C = (c_1, c_2)$: compute $m = c_2 / c_1^A$.

NOTE 9.2.2

The key g^B (called ephemeral) should not be reused. Thus, Elgamal is a probabilistic scheme.

Observe that Elgamal cipher is based in DH key agreement to get a shared key $\mathsf{k} = g^{A \cdot B}$. Then k is used to mask the message

$$\mathsf{Enc}_{\mathsf{k}}(m) = m \cdot \mathsf{k} \pmod p.$$

Exercise 9.2.3

What are the plaintext space $\mathcal{P}$ and the ciphertext space $\mathcal{C}$ in the Elgamal cipher ?

Exercise 9.2.4

Consider **Elgamal** with $p = 83$ and $g = 4$. Encipher $m = (011101)_2$ with $A = 37$.

Exercise 9.2.5

Two messages m_1, m_2 where enciphered by using the same ephemeral key g^B. If the cryptanalyst knows or guess one message then it can read the other.

Attacks

active: MITM.

Malleability: replace $C = (c_1, s \cdot c_2)$ where s is chosen by the cryptanalyst. Notice that the decrypted message is going to be $s \cdot m$. Thus some padding should be introduced.

passive: by solving DLP.

Trying to find reused ephemeral keys with known/guess messages.

IND-CPA: If the generator $g = r^2$ is a quadratic residue then the c_2 is a quadratic residue if and only if m is so. Thus, the message space must be reduced (to all quadratic residues) to achieve IND-CPA security.

9.2.2 Rabin cryptosystem

9.2.6 Rabin textbook

- $\mathsf{Gen}(\lambda)$: choose $p, q \equiv 3 \pmod 4$ (prime numbers of λ bit) , compute $N = pq$.
- Set $\mathsf{pk} = N$ and $\mathsf{sk} = (p, q)$.
- $\mathsf{Enc}_{\mathsf{pk}}(m)$ with $m \in \mathbb{Z}_N$: compute $C = m^2 \bmod N$.
- $\mathsf{Dec}_{\mathsf{sk}}(C)$: compute $m_p = C^{\frac{p+1}{4}} \bmod p$, $m_q = C^{\frac{p+1}{4}} \bmod q$.

  CRT gives four candidates for m: $(\pm m_p, \pm m_q)$.

  So some redundance/special structure/padding should be added to ensure decryption. In [Rabin79] say nothing about how to ensure decryption. See the appendix for some ways to do so.

Exercise 9.2.7

Let $p = 7, q = 11$ and $C = 23$. Find the four candidates to be the message m.

Exercise 9.2.8

As before let $p = 7, q = 11$ and $C = 23$ and assume that m itself must be a quadratic residue modulo 77. Find the message m.

NOTE 9.2.9

It has been proven that any algorithm which decrypts a Rabin-encrypted value can be used to factor the modulus N.

9.2.3 RSA

9.2.10 RSA textbook

- $\mathsf{Gen}(\lambda)$: choose p, q (prime numbers of λ bit), compute $N = pq$ and $\phi(N) = (p-1)(q-1)$. Pick $e \in \mathbb{Z}^*_{\phi(N)}$ and compute $d = e^{-1} \bmod \phi(N)$. Set $\mathsf{pk} = (N, e)$ and $\mathsf{sk} = (\phi(N), d)$.
- $\mathsf{Enc}_{\mathsf{pk}}(m)$ with $m \in \mathbb{Z}_N$: compute $C = m^e \bmod N$.
- $\mathsf{Dec}_{\mathsf{sk}}(C)$: compute $m = C^d \bmod N$.

Notice that

$$x^2 + (\phi(N) - N - 1)x + N = (x-p)(x-q)$$

so knowledge of N and $\phi(N)$ is computationally equivalent to knowledge of both prime numbers p and q.
It is possible recover p, q from e, d, N see page 111 in Special Publication (NIST SP) - 800-56B Rev. 2

Exercise 9.2.11

Consider textbook RSA as above with $p = 11$, $q = 23$ ed $e = 3$. Encipher $m = 0111001$ (string of bits) and find the secret key $\mathsf{sk} = (\phi(N), d)$. Decipher $C = 11010101$.

Exercise 9.2.12

A company wants to use textbook RSA with the same N to encipher emails of their employers. To ensure privacy different exponents are distributed between the employers. Namely employer i has public key (N, e_i) and secret key $(\phi(N), d_i,)$.

Is this a good idea? In particular:

1. could employer i decipher emails enciphered with the public key of some colleague?
2. what happens if an eavesdropper got the same message m encipher by the public keys of two employers ? (i.e. enciphered with (N, e_i) and with (N, e_j) where $e_i \neq e_j$)?

NOTE 9.2.13

$e = 3$ is usually used but keep mind that there are attacks, see next exercise. In practice the involved prime numbers are of 2048 bits. Other frequent values used as e are 17 and $65537 = 2^{16} + 1$ for speed-up encryption purposes.

Exercise 9.2.14

Assume that message m is encrypted with $e = 3$ with respect to three RSA modulus N_1, N_2, N_3. That is to say, you do

$$\begin{cases} C_1 = m^3 \pmod{N_1} \\ C_2 = m^3 \pmod{N_2} \\ C_3 = m^3 \pmod{N_3} \end{cases}$$

Show that a cryptanalyst can recover m from $C_1, C_2, C_3, N_1, N_2, N_3$.
Hint 1: Notice that $m \leq N_1$, $m \leq N_2$ and $m \leq N_3$. Use CRT and that $m^3 \leq N_1 N_2 N_3$.
Hint 2: `https://www.johndcook.com/blog/2019/03/06/rsa-exponent-3/`

NOTE 9.2.15

The obvious way to attack RSA is factorization of N. But it is a conjecture that to compute m from C and the public key (N, e) it is necessary to factor N.

attacks

Mathematical attacks: Factorization methods improved considerably during the last years. So for long-term security 2048-4096 bits long key are recommended.

Pollard's rho: $O(n^{\frac{1}{4}})$
elliptic-curve factorization: $e^{(1+o(1))\sqrt{\log(n)(\log\log(n))}}$

Wiener attacks for small d.

Side-Channel: see [Paar10, page 195].

Protocol attacks: exploit weaknesses in the way RSA is being used. There have been several protocol attacks over the years. Among the better known ones are the attacks that exploit the malleability of RSA, which was introduced in the previous section. Many of them can be avoided by using padding. Modern security standards describe exactly how RSA should be used, and if one follows those guidelines, protocol attacks should not be possible.

9.2.16 Chosen Ciphertext Attack (CCA)

Let (N, e) be a RSA public key and let $C = m^e$ a ciphertext. To get m pick a random r and ask the owner of (N, e) to decipher the following chosen ciphertext :

$$r^e \cdot C$$

So you got $r \cdot C^d$ hence $r \cdot m$ ☺

9.2.4 padded RSA

The problem with RSA textbook is that is not CPA-IND secure. Indeed, this is so as in any deterministic cypher.

To get a probabilistic RSA the message space is restricted to strings m of l bits with $l = 2\lambda - r$. Then to encrypt a message m a random string s of r bits is picked and

$$C = \mathsf{Enc}_{\mathsf{pk}}(m) = (s||m)^e \pmod N$$

To decrypt we compute $C^d \pmod N$ and discard the first r bits to get m.

NOTE 9.2.17

PKCS stands for "Public Key Cryptography Standards". These are a group of public-key cryptography standards devised and published by RSA Security LLC, starting in the early 1990s. `https://en.wikipedia.org/wiki/PKCS`

9.3 Hybrid Encryption, KEM/DEM Paradigm

The basic idea is to use public-key encryption to obtain a shared key k, and then encrypt the message m using a private-key encryption scheme and key k. The receiver uses its long-term (asymmetric) private key to derive k, and then uses private-key decryption (with key k) to recover the original message.

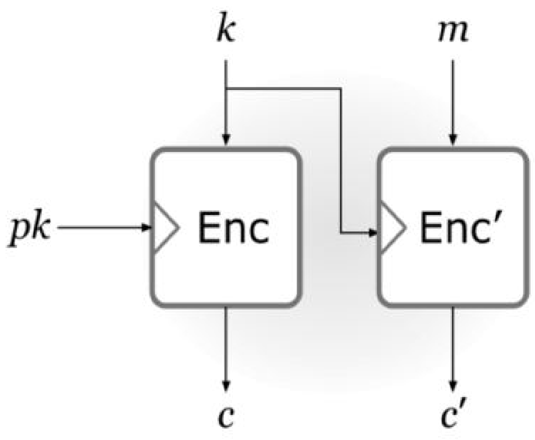


Figure 1: Hybrid Encryption

A more direct approach is to use a public-key primitive called a key-encapsulation mechanism (KEM) to accomplish both of these in one shot. This is advantageous both from a conceptual point of view and in terms of efficiency.

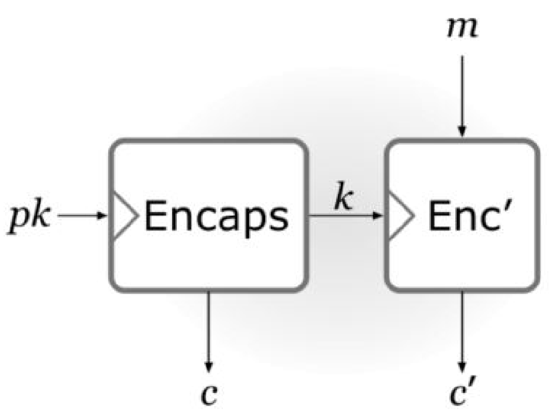


Figure 2: KEM and Data-Encryption Mechanism (DEM)

9.3.1 Key-Encapsulation Mechanism (KEM)

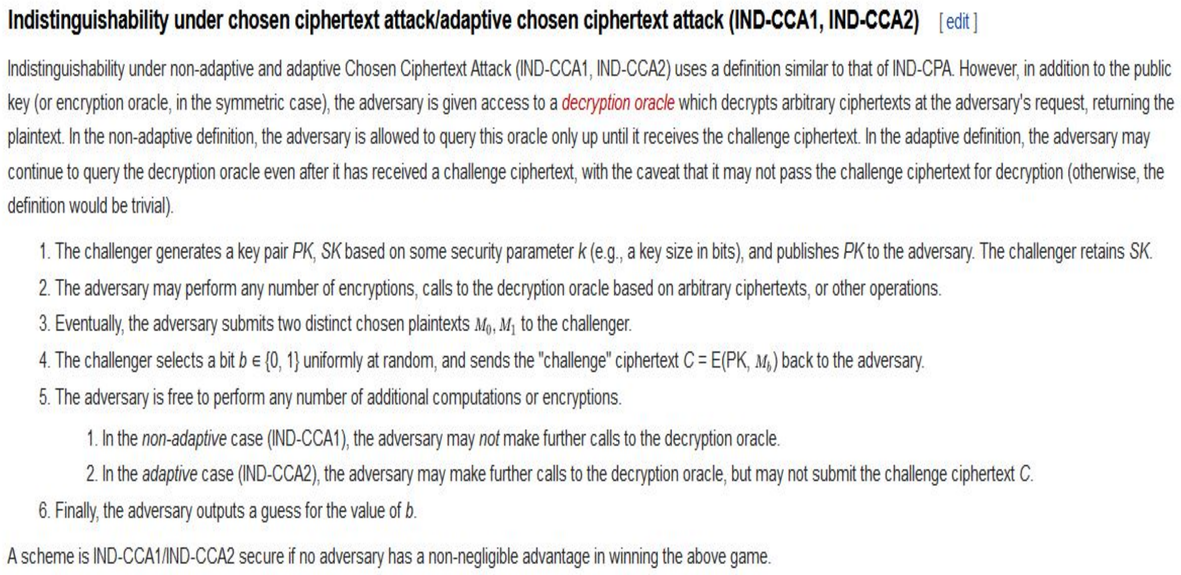
is a tuple of three algorithms $(\mathsf{Gen}, \mathsf{Encaps}, \mathsf{Decaps})$:

(i) the key generator algorithm Gen takes as input a security parameter 1^n and outputs a public-/private-key pair $(\mathsf{pk}, \mathsf{sk})$ each of length at least n.

(ii) the encapsulation algorithm Encaps takes as input a public key pk and the security parameter 1^n. It outputs a ciphertext c and a key $\mathsf{k} \in \{0,1\}^{l(n)}$ where l is the key length. This is written as $(c, \mathsf{k}) \leftarrow \mathsf{Encaps}_{\mathsf{pk}}(1^n)$

(iii) the decapsulation algorithm Decaps takes as input a private key sk and a ciphertext c, and outputs a key k or a special symbol $\perp$ denoting failure. This is written as $\mathsf{k} = \mathsf{Decaps}_{\mathsf{sk}}(c)$.

9.3.1 CCA-IND and the Fujisaki-Okamoto transformation



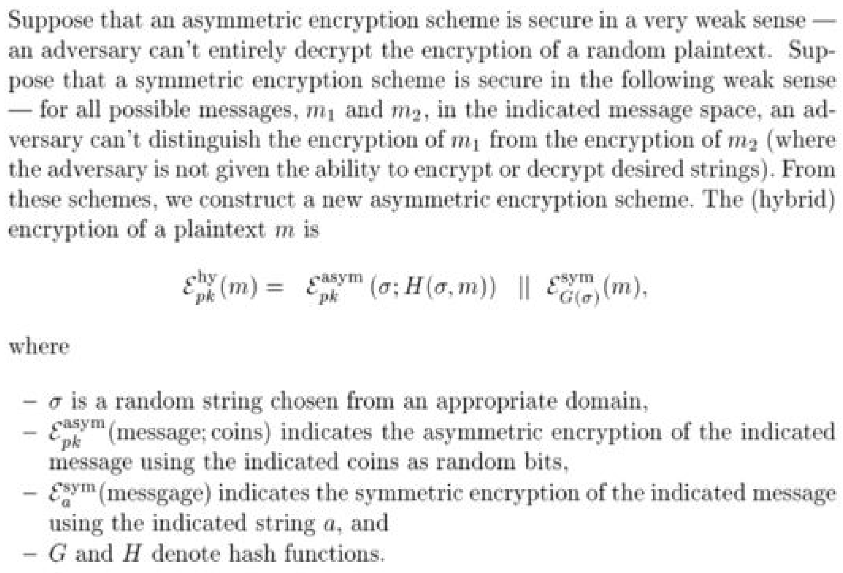
Indistinguishability under chosen ciphertext attack/adaptive chosen ciphertext attack (IND-CCA1, IND-CCA2) [edit]

Indistinguishability under non-adaptive and adaptive Chosen Ciphertext Attack (IND-CCA1, IND-CCA2) uses a definition similar to that of IND-CPA. However, in addition to the public key (or encryption oracle, in the symmetric case), the adversary is given access to a decryption oracle which decrypts arbitrary ciphertexts at the adversary's request, returning the plaintext. In the non-adaptive definition, the adversary is allowed to query this oracle only up until it receives the challenge ciphertext. In the adaptive definition, the adversary may continue to query the decryption oracle even after it has received a challenge ciphertext, with the caveat that it may not pass the challenge ciphertext for decryption (otherwise, the definition would be trivial).

1. The challenger generates a key pair *PK*, *SK* based on some security parameter *k* (e.g., a key size in bits), and publishes *PK* to the adversary. The challenger retains *SK*.
2. The adversary may perform any number of encryptions, calls to the decryption oracle based on arbitrary ciphertexts, or other operations.
3. Eventually, the adversary submits two distinct chosen plaintexts M_0, M_1 to the challenger.
4. The challenger selects a bit $b \in \{0, 1\}$ uniformly at random, and sends the "challenge" ciphertext $C = E(PK, M_b)$ back to the adversary.
5. The adversary is free to perform any number of additional computations or encryptions.
    1. In the *non-adaptive* case (IND-CCA1), the adversary may *not* make further calls to the decryption oracle.
    2. In the *adaptive* case (IND-CCA2), the adversary may make further calls to the decryption oracle, but may not submit the challenge ciphertext *C*.
6. Finally, the adversary outputs a guess for the value of *b*.

A scheme is IND-CCA1/IND-CCA2 secure if no adversary has a non-negligible advantage in winning the above game.

Figure 3: CCA-IND



Suppose that an asymmetric encryption scheme is secure in a very weak sense — an adversary can't entirely decrypt the encryption of a random plaintext. Suppose that a symmetric encryption scheme is secure in the following weak sense — for all possible messages, m_1 and m_2, in the indicated message space, an adversary can't distinguish the encryption of m_1 from the encryption of m_2 (where the adversary is not given the ability to encrypt or decrypt desired strings). From these schemes, we construct a new asymmetric encryption scheme. The (hybrid) encryption of a plaintext m is

$$\mathcal{E}^{\mathrm{hy}}_{pk}(m) = \mathcal{E}^{\mathrm{asym}}_{pk}(\sigma; H(\sigma, m)) \;\;||\;\; \mathcal{E}^{\mathrm{sym}}_{G(\sigma)}(m),$$

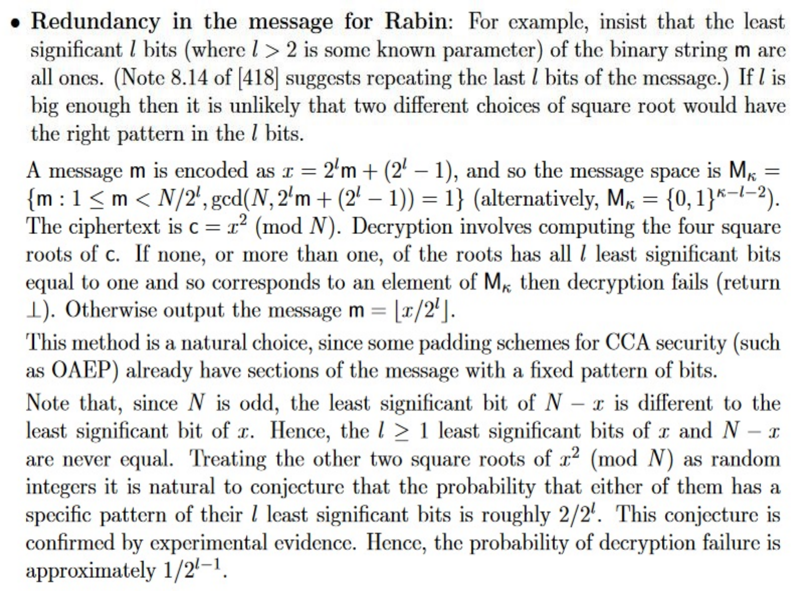
where

- σ is a random string chosen from an appropriate domain,
- $\mathcal{E}^{\mathrm{asym}}_{pk}$(message; coins) indicates the asymmetric encryption of the indicated message using the indicated coins as random bits,
- $\mathcal{E}^{\mathrm{sym}}_{a}$(messgage) indicates the symmetric encryption of the indicated message using the indicated string a, and
- G and H denote hash functions.

Figure 4: Fujisaki-Okamoto scheme

9.4 Appendix

9.4.1 Rabin decryption



- **Redundancy in the message for Rabin**: For example, insist that the least significant l bits (where $l > 2$ is some known parameter) of the binary string m are all ones. (Note 8.14 of [418] suggests repeating the last l bits of the message.) If l is big enough then it is unlikely that two different choices of square root would have the right pattern in the l bits.

A message m is encoded as $x = 2^l\mathsf{m} + (2^l - 1)$, and so the message space is $\mathsf{M}_\kappa = \{\mathsf{m} : 1 \le \mathsf{m} < N/2^l, \gcd(N, 2^l\mathsf{m} + (2^l - 1)) = 1\}$ (alternatively, $\mathsf{M}_\kappa = \{0,1\}^{\kappa-l-2}$). The ciphertext is $\mathsf{c} = x^2 \pmod{N}$. Decryption involves computing the four square roots of c. If none, or more than one, of the roots has all l least significant bits equal to one and so corresponds to an element of M_κ then decryption fails (return $\perp$). Otherwise output the message $\mathsf{m} = \lfloor x/2^l \rfloor$.

This method is a natural choice, since some padding schemes for CCA security (such as OAEP) already have sections of the message with a fixed pattern of bits.

Note that, since N is odd, the least significant bit of $N - x$ is different to the least significant bit of x. Hence, the $l \ge 1$ least significant bits of x and $N - x$ are never equal. Treating the other two square roots of $x^2 \pmod{N}$ as random integers it is natural to conjecture that the probability that either of them has a specific pattern of their l least significant bits is roughly $2/2^l$. This conjecture is confirmed by experimental evidence. Hence, the probability of decryption failure is approximately $1/2^{l-1}$.

Figure 5: Rabin redundance in encryption [Gal18, Chapter 24, 2.1, pag.514]

9.5 Bibliography

Books I used to prepare this note:

[Gal18] Steven Galbraith, *Mathematics of Public Key Cryptography*, version 2.0 https://www.math.auckland.ac.nz/~sgal018/crypto-book/main.pdf

[KatLin15] Jonathan Katz; Yehuda Lindell, *Introduction to Modern Cryptography* Second Edition, Chapman & Hall/CRC, Taylor & Francis Group, 2015.

[Paar10] Paar, Christof, Pelzl, Jan, *Understanding Cryptography, A Textbook for Students and Practitioners*, Springer-Verlag, 2010.

[Schneier15] Bruce Schneier, *Applied Cryptography: Protocols, Algorithms and Source Code in C* ,Wiley; 20th Anniversary edition,2015.

Here a list of papers:

[BJN00] Boneh D., Joux A., Nguyen P.Q. *Why Textbook ElGamal and RSA Encryption Are Insecure.* In: Okamoto T. (eds) Advances in Cryptology ASIACRYPT 2000. ASIACRYPT 2000. Lecture Notes in Computer Science, vol 1976. Springer, Berlin, Heidelberg (2000) https://link.springer.com/chapter/10.1007/3-540-44448-3_3

[DH76] Diffie, W.; Hellman, M. *New directions in cryptography*, (1976). IEEE Transactions on Information Theory. 22 (6): 644654. https://ee.stanford.edu/%7Ehellman/publications/24.pdf

[Elgamal85] Taher Elgamal *A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms*, (1985). IEEE Transactions on Information Theory. 31 (4): 469472. https://caislab.kaist.ac.kr/lecture/2010/spring/cs548/basic/B02.pdf

[FO99] Fujisaki, E.; Okamoto, T. *Secure Integration of Asymmetric and Symmetric Encryption Schemes*, in Advances in CryptologyCRYPTO99, ed. by M. Wiener. Lecture Notes in Computer Science, vol. 1666(Springer, Berlin, 1999), pp. 537554. https://link.springer.com/chapter/10.1007/3-540-48405-1_34

[Rabin79] M. Rabin, *Digitalized Signatures and Public-Key Functions as Intractable as Factorization*, MIT Laboratory for Computer Science, January 1979. http://publications.csail.mit.edu/lcs/pubs/pdf/MIT-LCS-TR-212.pdf

and some interesting links:

Integer Factoring, by A.K. Lenstra

https://en.wikipedia.org/wiki/OpenSSL

https://www.math.auckland.ac.nz/~sgal018/crypto-book/ch24.pdf