CCA - Security

<u>THM</u> : Cramer - Shoup PKE is CCA-secure under DDH.

Proof: We consider some hybrid.

$\text{GAME}^{CCA}_{\pi,A}$

$A(1^\lambda)$ $\qquad\qquad$ $C(1^\lambda)$

$\xleftarrow{\quad pk \quad}$

$\xrightarrow{c = (c_1, c_2, c_3, c_4)}$ $\quad (G, g, q) \leftarrow \$ \text{GroupGen}(1^\lambda)$

$\xleftarrow{(m \text{ or } \perp)}$ $\quad g_h = g_1 g_2 = g_1^{\alpha}$

g

$\xrightarrow{m_0^*, m_1^*}$ $\quad \text{params} = (G, g_1, g_2, q)$

$\xleftarrow{\quad c^* \quad}$ $\quad x_1, x_2, x_3 \leftarrow \$ \mathbb{Z}_q$

$\xrightarrow{c \neq c^*}$ $\quad sk = \begin{pmatrix} x_1, x_2, x_3 \\ y_1, y_2, y_3 \end{pmatrix} \leftarrow \$ \mathbb{Z}_q$

$\xleftarrow{m \text{ or } \perp}$

$\xrightarrow{\quad b' \quad}$ $\quad pk = (\text{params}, h_1, h_2, h_3)$

$$h_1 = g_1^{x_1} g_2^{y_1}, \quad h_2 = g_1^{x_2} g_2^{y_2}, \ldots$$

$$g^3 = g_1^r \quad g_4 = g_2^r \quad r \leftarrow \$ \mathbb{Z}_q$$

$$c_1^* = g_3 \quad c_2^* = g_4$$

$$c_3^* = g_3^{x_1} \cdot g_4^{y_1} \cdot m_b^*$$

$$c_4^* = g_3^{x_2 + \beta x_3} \; g_4^{x_3 + \beta y_2}$$

y2+betay3

$\mathrm{HYB}^{\mathrm{CCA}}_{\Pi,A}$

$$
\begin{array}{lcl}
A(1^\lambda) & & C(1^\lambda) \\
 & \xleftarrow{\quad pk \quad} & \\
 & \xrightarrow{\quad c = (c_1, c_2, c_3, c_4) \quad} & (\mathbb{G}, g, q) \leftarrow \$ \mathrm{GroupGen}(1^\lambda) \\
 & \xleftarrow{\quad (m \text{ or } \perp) \quad} & g_h = g_1 g_2 = g_1^{\alpha} \\
 & \xrightarrow{\quad m_0^*, m_1^* \quad} & \mathrm{params} := (\mathbb{G}, g_1, g_2, q) \\
 & \xleftarrow{\quad c^* \quad} & x_1, x_2, x_3 \leftarrow \$ \mathbb{Z}_q \\
 & \xrightarrow{\quad c \neq c^* \quad} & sk = \begin{pmatrix} x_1, x_2, x_3 \\ y_1, y_2, y_3 \end{pmatrix} \leftarrow \$ \mathbb{Z}_q \\
 & \xleftarrow{\quad m \text{ or } \perp \quad} & \\
 & & pk = (\mathrm{params}, h_1, h_2, h_3) \\
 & \xrightarrow{\quad b' \quad} & \\
 & & h_1 = g_1^{x_1} g_2^{y_1}, \; h_2 = g_1^{x_2} g_2^{y_2}, \dots
\end{array}
$$

$g_3 = g_1^{r} \quad g_4 = g_2^{r'}$

$r, r' \leftarrow \$ \mathbb{Z}_q$

$c_1^* = g_3 \quad c_2^* = g_4$

$c_3^* = g_3^{x_1} \cdot g_4^{y_1} \cdot m_b^*$

$c_4^* = g_3^{x_2 + \beta x_3} g_4^{x_3 + \beta y_2}$

y2+betay3

LEMMA: $\mathrm{GAME}(\lambda, b) \approx_c \mathrm{HYB}(\lambda, b)$, $\forall b \in \{0,1\}$

Excercise by DDH

LEMMA: $\mathrm{HYB}(\lambda, 0) \approx_c \mathrm{HYB}(\lambda, 1)$

Proof (sketch): Similar to cs-lite. In particular, it still holds that so long as $\mathcal{A}$ makes no ILLEGAL decryption query c that is not rejected, then b is information-theoretically hidden (Try as excercise)

<u>CLAIM</u> Attacker can make decryption query c that is ILLEGAL and not rejected only with negligible prob.

Proof: What does $\mathcal{A}$ know about x_2, y_2, x_3, y_3?

- $\log_{g_1} h_2 = x_2 + \alpha y_2$
- $\log_{g_1} h_3 = x_3 + \alpha y_3$ $\alpha = \log_{g_1} g_2$

Given the challenge $c^* = (g_3, g_4, c_3^* = g_3^{x_1} g_4^{y_1} m_b, c_4^*)$

with $g_3 = g_1^r$, $g_4 = g_2^{r'}$ for $r \neq r'$ (whp). $\beta = H(c_1, c_2, c_3)$

- $\log_{g_1} c_4^* = (x_2 + \beta y_2) r + (x_3 + \beta y_3) \alpha r'$

because $c_4^* = g_3^{x_2 + \beta y_2} \cdot g_4^{x_3 + \beta y_3}$ (the attacker knows these 3 equation)

Let $c = (c_1, c_2, c_3, c_4)$ $\neq c^*$ be any decryption query
Look at cases:

1. $(c_1, c_2, c_3) = (c_1^*, c_2^*, c_3^*)$, but $c_4 \neq c_4^*$

Then, $H(c_1, c_2, c_3) = H(c_1^*, c_2^*, c_3^*) = \beta$

$$c_1^{x_2 + \beta x_3} \cdot c_2^{y_2 + \beta y_3} = c_1^{*\, x_2 + \beta x_3} \cdot c_2^{*\, y_2 + \beta y_3}$$

$= c_4^* \neq c_4$

So these queries are ALWAYS REJECTED

2. $(c_1, c_2, c_3) \neq (c_1^*, c_2^*, c_3^*)$, but $H(c_1, c_2, c_3) = H(c_1^*, c_2^*, c_3^*)$

This happens with $p = negl(\lambda)$ by collision resistance of H

3. $(c_1, c_2, c_3) \neq (c_1^*, c_2^*, c_3^*)$, $H(c_1, c_2, c_3) = \beta \neq H(c_1^*, c_2^*, c_3^*)$

In order for c_4 not to be rejected we need

$$\log_{g_1} c_4 = (x_2 + \beta x_3) r_1 + (y_2 + \beta y_3) \alpha r_2$$

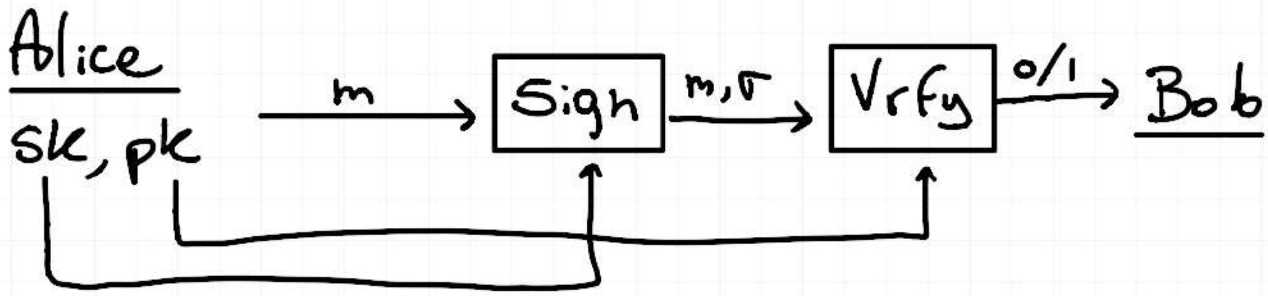
$$\log_{g_1} c_1 = r_1 \neq r_2 = \log_{g_2} c_2$$

è diverso

Fact: So long as $\beta = \beta^*$, $r_2 \neq r_1$, $r \neq r'$, the above equation is linearly independent of the 3 previous equations. The system has a unique solution, uniformly likely

As in CS-lite, this implies the claim. □

DIGITAL SIGNATURE



We have two more algorithms, and there is no way that Bob can compute the signature without explicitly knowing Alice's sk. This kind of technology is used for example in Bitcoin.

In practice we use DIGITAL SIGNATURES to authenticate public keys. By using PUBLIC-KEY INFRASTRUCTURE.

Alice — KEY EXCHANGE — Amazon

Why does Alice believe that pk is from Amazon?

$\xleftarrow{\quad pk \quad}$ (pk, sk) for PKE

$K \leftarrow \$ \{0,1\}^\lambda$ for AES

$c \leftarrow Enc(pk, k)$ $\xrightarrow{\quad c \quad}$ $k = Dec(sk, c)$

SECURE CHANNEL

This method is the base of the TLS Protocol

$$cert_{CA \to AM} = Sign(sk_{CA}, pk \| Amazon)$$

Amazon — CA

$\xrightarrow{\quad pk \quad}$ pk_{ca}, sk_{ca}

$\xleftarrow{\quad cert_{CA \to AM} \quad}$

Alice checks $Vrfy(pk_{ca}, pk \| Amazon, cert_{CA \to AM}) = 1$

But why does Alice believe pk_{ca} is actually from CA?

Security: UF-CMA $\rightarrow$ $\mathrm{GAME}^{\mathrm{ufcma}}_{\Pi,\mathcal{A}}(\lambda)$

$\mathcal{A}(1^\lambda)$ $\qquad\qquad\qquad\qquad$ $\mathcal{C}(1^\lambda)$

$$\xleftarrow{\quad pk \quad} \qquad (pk, sk) \leftarrow\$ \mathrm{KGen}(1^\lambda)$$

$$\xrightarrow{\quad m \quad}$$

$$\xleftarrow{\quad \sigma \quad} \qquad \sigma = \mathrm{Sign}(sk, m)$$

$$\xrightarrow{\quad m^*, \sigma^* \quad} \qquad A \text{ wins if}$$

$\mathrm{Vrfy}(pk, m^*, \sigma^*) = 1$

$m^* \neq \{m\}$

THM: UF-CMA signatures exist assuming OWFs.
But it's not practical...

What about RSA? Would this work?

$$(n, e) = pk \qquad d = sk$$

$$(n, e, d) \leftarrow \$ \mathrm{GenModulus}(1^\lambda)$$

$$\mathrm{Sign}(m, sk) = m^d \bmod n$$

$$\mathrm{Vrfy}(pk, m, \sigma) = \sigma^e \bmod n = m \bmod n$$

Correctness by Fermat: $\sigma^e = (m^d)^e = m^{ed}$

$$= m^{t \cdot \varphi(n) + 1}$$

$$= m \bmod n \quad \checkmark$$

UF-CMA? Assume given (m,σ), (m',σ')

Forge for $m \cdot m' = m^*$

$\sigma \cdot \sigma' = \sigma^*$

Also, without sign. queries: pick $\sigma \in \mathbb{Z}_n^*$

Let $m = \sigma^e \bmod n$

Forge m, σ

How to fix this? Hash the message!

$$\mathrm{Sign}(sk, m) = H(m)^d \bmod n$$

$$\mathrm{Vrfy}(pk, m, \sigma) = \sigma^e \stackrel{?}{=} H(m)$$

Why is this secure? Intuitively we need CR: given valid (m,σ), if I can find $m' \neq m$ with $H(m) = H(m')$, then (m',σ) is also valid.

Let's abstract it: RSA is just a TDP:

(Gen, f, f') f^(-1)

$\mathrm{KGen}(1^\lambda) = \mathrm{Gen}(1^\lambda) \xrightarrow{\$} (pk, sk)$

$\mathrm{Sign}(m, sk) = f^{-1}(sk, H(m))$ sk

$\mathrm{Vrfy}(pk, m, \sigma): f(pk, \sigma) \stackrel{?}{=} H(m)$

FULL DOMAIN HASH (it's used in real life)

This works, only assuming H behaves like a RANDOM

ORACLE, and it is PROVABLY SECURE

$$
\begin{array}{lcl}
A(1^\lambda) & & \mathcal{C}(1^\lambda) \\
 & \xrightarrow{\quad x \quad} & \text{If } x \text{ not asked before} \\
 & \xleftarrow{\quad h \quad} & \quad h \leftarrow \$ \, Im(H) \\
 & & \quad \text{store } (x,h) \\
 & & \text{else return } h
\end{array}
$$

The RANDOM ORACLE methodology: Assume algorithm and attacker have access to $H(\cdot)$.

Why this? Clearly security is only heuristic, because sometimes it's impossible to do things without a random oracle. Also, it is super efficient when replacing RO with SHA-3.

<u>THM</u>: Full Domain Hash is UF-CMA in the RO model assuming (Gen, f, f^{-1}) is a TDP.

Proof: We need to show that no A PPT exist s.t.

$$\mathrm{GAME}^{UFCMA}_{\Pi,A}(\lambda) \qquad \text{RO } H: \{0,1\}^\lambda \to \mathcal{X}_{pk}$$

$$
\begin{array}{lcl}
A(1^\lambda) & & \mathcal{C}(1^\lambda) \\
 & \xleftarrow{\quad pk \quad} & (pk,sk) \leftarrow \$ \, Gen(1^\lambda) \\
 & \xrightarrow{\quad m \quad} & \\
 & \xleftarrow{\quad H(m) \quad} &
\end{array}
$$

RO QUERIES

sign queries

$$\xrightarrow{m} \quad \sigma = f^{-1}(sk, H(m))$$
$$\xleftarrow{\sigma}$$

$$\xrightarrow{(m^*, \sigma^*)} \quad A \text{ wins if}$$

$f(pk, \sigma^*) = H(m^*)$

m^* is FRESH

Assume $\exists$ PPT attacker A succeeding up $\geq 1/poly$ and construct A' for TDP.

Assumption: Before signature query on m_i (or forgery on m^*), attacker asks m_i to RO (or m^*).
Also A never repeats queries.

Try to guess query m^* to RO

$A(1^\lambda) \xleftarrow{pk} A' \xleftarrow{(pk, y)} C$

$(pk, sk) \leftarrow\$ Gen(1^\lambda)$

$x \leftarrow \mathcal{X}_{pk}$

$y = f(pk, x)$

RO QUERIES

$\xrightarrow{m_i}$ $j \leftarrow\$ [q_h]$

$\xleftarrow{y_i}$ If $i \neq j$

$x_i \leftarrow \mathcal{X}_{pk}$

$y_i = f(pk, x_i)$

Else return $y = y_i$

↳ RO programming

perfect simulation as

$$\{\hat{y} \leftarrow\$ \mathcal{X}_{pk}\} \equiv \left\{\hat{y} : \begin{array}{l}\hat{x} \leftarrow \mathcal{X}_{pk} \\ \hat{y} \leftarrow f(pk, \hat{x})\end{array}\right\}$$

$\xrightarrow{m_i}$ Perfect simulation as

$\xleftarrow{x_i = \sigma_i}$ $x_i = f^{-1}(sk, y_i)$

$$= f^{-1}(sk, H(m_i))$$

$$(m^*, \sigma^*) \longrightarrow \text{IF } m^* = m_j \rightarrow \xrightarrow{\sigma^*} A' \text{ wins because}$$

Else ABORT

$$\sigma^* = f^{-1}(sk, H(m^*))$$
$$= f^{-1}(sk, y)$$

$$Pr[A' \text{ wins}] = Pr[A \text{ wins} \wedge A' \text{ guesses } j]$$

$$= Pr[A' \text{guesses } j] \cdot Pr[A \text{ wins} \mid A' \text{guesses } j]$$

$$\geq 1/q_h \cdot 1/poly = 1/poly$$

The power of ROs.

Let RO: $\{0,1\}^* \rightarrow \{0,1\}^*$ be a RO. Then

↗ collision resistant hash function

1. $H(x) = RO(x)$ is a CRH

$$Pr[H(x) = H(x') : (x, x') \leftarrow \$ A^{RO(\cdot)}(1^\lambda)]$$

$$= Pr[\exists x_i \neq x_j : H(x_i) = H(x_j) \text{ for the queries } x_1, \dots x_q]$$

$$\leq \binom{q}{2} \cdot 2^{-n} \leq q^2 \cdot 2^{-n}$$ n = bit-size of x

$$= negl(\lambda) \quad \text{for } n = \omega \log \lambda$$
$$q = poly(\lambda)$$

2. $G^{RO}(x) = RO(x \| 0) \| RO(x \| 1)$ is a PRG

3. $F^{RO}(x) = RO(k \| x)$ is a PRF

4. CCA-2 PKE: OAEP (PKCS #2) from RSA