Cryptography 2021

Valerio Venanzi

December 25, 2021

Contents

Perfect Secrecy

A ciphertext is perfectly secret if it reveals nothing about the plaintext.

Let M be the space of messages (plaintext), C be the space of ciphertexts and k a key.
$\forall m \in M$, $\forall c \in C$, let $C = Enc(k, M)$

$$Pr[M = m] = Pr[M = m|C = c]$$

Some equivalent definitions of perfect secrecy:

- $Pr[M = m] = Pr[M = m|C = c]$
- M and C are independent
- $\forall m, m' \in \mathcal{M}$, $\forall c \in C$ $Pr[Enc(k, m) = c] = Pr[Enc(k, m') = c]$

1.1 One Time Pad - OTP

Let $k, m, c \in \{0, 1\}^\lambda$

$Enc(k, m) = c = k \oplus m$
$Dec(k, c) = c \oplus k = (k \oplus m) \oplus k = m$

Theorem 1.1.1. *Above OTP is perfectly secret.*

Proof.
$Pr[Enc(k, m) = c] = Pr[k \oplus m = c] = Pr[k = c \oplus m] = 2^{-\lambda}$
$Pr[Enc(k, m') = c] = Pr[k \oplus m' = c] = Pr[k = c \oplus m'] = 2^{-\lambda}$ □

Limitations:

- $|k| = |m|$
- One time

Computational Security

2.1 Polynomial

A function $p : \mathbb{N} \to \mathbb{N}$ is polynomial in λ (i.e. $p(\lambda) = poly(\lambda)$) if:
$\exists c \geq 0$ s.t. $p(\lambda) = O(\lambda^c) \longrightarrow \exists c, c', \lambda_0 \in \mathbb{N}$ s.t. $\lambda \geq \lambda_0 \; p(\lambda) \leq c' \cdot \lambda^c$.

A function that runs in polynomial time is said to be efficient.

2.2 Negligible

A function $\varepsilon : \mathbb{N} \to [0, 1]$ is negligible if:
$\varepsilon(\lambda) = O(\frac{1}{poly(\lambda)})$

2.3 Probabilistic Polynomial Time Turing Machine - PPT

A Turing Machine - TM A is PPT if it's worst case's running time is polynomial in the input length.
For instance: $\exists p(\lambda) = poly(\lambda)$ s.t. $\forall x \in \{0,1\}^*$, $\forall r \in \{0,1\}^*$ (r is a **coin**, it represents randomness), $A(x, r)$ terminates after $p(\lambda)$ steps.

2.4 One-Way Function - OWF

A deterministic function $p : \{0,1\}^* \to \{0,1\}^*$ is s OWF if it's difficult to invert.

$\forall$PPT A, we have that $Pr[A\ wins] = negl(\lambda)$ in the following game:

1. C picks a random $x \leftarrow\$ \{0,1\}^*$
2. C calculates $y = f(x)$
3. C sends y to A
4. A sends x' to C

A wins if $f(x') = y$, because A managed to invert f.

Pseudorandomness

3.1 Pseudo Random Generator - PRG

A deterministic function $G : \{0,1\}^\lambda \to \{0,1\}^{\lambda+l}$ is a PRG with stretch $l = l(\lambda)$ if:

- G if poly-time computable
- $l(\lambda) \geq 1$
- $G(U_\lambda) \approx_c U_{\lambda+l}$

Where U_λ is the space of uniformely distributed keys of length λ ($U_{\lambda+l}$ accordingly).

3.2 Computationally close

Simply put, if two things are computationally close it means that we can distinguish them only with negligible probability.
For instance:

$REAL_{G,A}(\lambda)$

1. C picks $s \leftarrow\$ \{0,1\}^\lambda$
2. C calculates $z = G(s)$
3. C sends z to A
4. A sends back $b' \in \{0,1\}$

$RAND_{G,A}(\lambda)$
Exactly like $REAL$, except that s is not picked and z is calculated as $z \leftarrow\$ U_{\lambda+l}$.

b' represents whether A thinks that z comes from game $REAL$ or game $RAND$ (i.e. has calculated with G or has been sampled from $U_{\lambda+l}$).
If $REAL \approx_c RAND$ then the probability that A is right is negligible.

3.3 Hardcore predicate

Let $h : \{0,1\}^\lambda \to \{0,1\}$ be a polynomial time function.
Let $f : \{0,1\}^\lambda \to \{0,1\}^\lambda$ be a OWF.
Then h is a **hard core predicate** for f if:

$$(f(U_\lambda), h(U_\lambda)) \approx_c (f(U_\lambda), U_1)$$

3.4 CPA security

Chosen Plaintext Attack - CPA.
$\Pi = (Enc, Dec)$ is CPA secure if $GAME^{CPA}_{\Pi,A}(\lambda, 0) \approx_c GAME^{CPA}_{\Pi,A}(\lambda, 1)$

$GAME^{CPA}_{\Pi,A}(\lambda, b)$:

1. C samples a key $k \leftarrow\$ \mathcal{K}$
2. A sends a message m' and gets back from C a cipher c' for m' (can repeat poly-times)
3. A sends messages m_0, m_1 to C
4. C sends $c \leftarrow\$ Enc(k, m_b)$ to A (b either 0 or 1)
5. A sends a message m' and gets back from C a cipher c' for m' (can repeat poly-times)
6. A sends $b' \in \{0,1\}$

$GAME = 1$ (A wins) if $b' = b$; i.e. A understands whether C encrypted m_0 or m_1.

3.5 Pseudo Random Function families

It's a powerful generalization of PRGs.
Let $\mathcal{F} = \{f_k : \{0,1\}^m \to \{0,1\}^l\}$. $\mathcal{F}$ is a PRF family if $REAL_{\mathcal{F},A}(\lambda) \approx_c RAND_{\mathcal{F},A}(\lambda)$.

Shape of the two games:

1. A sends x to C and C responds with z (can repeat poly-times)
2. A sends b'

b' says whether A thinks the game currently played is $REAL$ or $RAND$.

In $REAL$ C samples a key $k \leftarrow\$ \{0,1\}^\lambda$ and calculates $z = f_k(x)$.

In $RAND$ C samples $R \leftarrow\$ \mathcal{R}(m, l, \lambda)$ and calculates $z = R(x)$.
$\mathcal{R}$ is a set of functions from $\{0,1\}^m$ to $\{0,1\}^l$. $|\mathcal{R}| = 2^{2^m \cdot l}$.
Note that R is deterministic, but randomly sampled from $\mathcal{R}$, so we can think of it as a random function (for the adversary the distribution of R and a random function should look the same).

$\boxed{\text{Is the very last thing correct?}}$

3.6 Message Authentication

Alice wants to send a message to Bob. To make sure that the message that Bob receives is the one sent by Alice they need some sort of authentication scheme.

Let $Tag : \mathcal{K} \times \mathcal{M} \to \tau$ a MAC.
Let k be a key and m a message. $Tag(k, m) = \tau$, τ is the authentication tag, calculated by Alice; when Bob will receive the message m' (supposedly $m' = m$) he will calculate $Tag(k, m') = \tau'$. If $\tau' = \tau$ then Bob knows that the message is the one sent by Alice.

3.7 Universal Forgeability CMA - UF CMA

Let $\Pi = Tag$ as in Message Authentication. Π if UF-CMA if $\forall$PPT A

$$Pr[GAME_{\Pi,A}^{UF-CMA}(\lambda) = 1] \leq negl(\lambda)$$

$GAME_{\Pi,A}^{UF-CMA}(\lambda)$:

1. C samples a key $k \leftarrow\$ \mathcal{K}$
2. A sends a message m to C and C replies with $\tau = Tag(k, m)$ (can repeat poly-times)
3. A sends a message m^* and a tag τ^*

A wins (i.e. $GAME = 1$) if $Tag(k, m^*) = \tau^*$ and $m^* \in \{m\}$.

Condition: A can't send $m^* = m$ in order to send $\tau^* = \tau$ and win the game.

3.8 Almost Universality - AU

Let $\mathcal{H} = \{h_s : \{0,1\}^n \to \{0,1\}^m\}_{s \in \{0,1\}^\lambda}$ be a family of **Collision Resistant Hard functions - CRH**.
$\mathcal{H}$ if $\varepsilon - AU$ if $\forall m, m' \in \{0,1\}^n$ $(m \neq m')$ we have that $Pr[h_s(m) = h_s(m')] \leq \varepsilon = negl(|s|)$.

3.9 Pseudo Random Permutation - PRP

$\mathcal{F} = \{f_k : \{0,1\}^n \to \{0,1\}^n\}$ is a PRP if $\forall x \in \{0,1\}^\lambda$ $\exists f_k^{-1}$ s.t.
$f_k^{-1}(f_k(x)) = x$.
Moreover $REAL_{\mathcal{F},A} \approx_c RAND_{\mathcal{F},A}$.

$REAL_{\mathcal{F},A}$

1. C samples a key $k \leftarrow\$ \mathcal{K}$
2. A sends a message m to C and C replies with $z = f_k(x)$ (can repeat poly-times)
3. A sends b'

$RAND_{\mathcal{F},A}$

1. C samples a permutation $P \leftarrow\$ \mathcal{P}(n, \lambda)$
2. A sends a message m to C and C replies with $z = P(x)$ (can repeat poly-times)
3. A sends b'

b' represents whether A thinks that z comes from $REAL$ or $RAND$.
$\mathcal{P}(n, \lambda)$ is the set of all possible permutations ovre $\{0,1\}^n$

3.10 Chosen Ciphertext Attack - CCA

CCA security is an enhancement of CPA, where the adversary can even send a ciphertext and see its plaintext (i.e. the adversary can decrypt). $\Pi = (Enc, Dec)$ is CCA-secure if $GAME^{CCA}_{\Pi,A}(\lambda, 0) \approx_c GAME^{CCA}_{\Pi,A}(\lambda, 1)$

$GAME^{CCA}_{\Pi,A}(\lambda, b)$:

1. C samples a key $k \leftarrow\$ \mathcal{K}$
2. A sends a message m_i to C and C sends back a ciphertext $c_i \leftarrow\$ Enc(k, m_i)$ (can repeat poly-times)
3. A sends a cipher c'_i to C and C sends back the plaintext $m'_i = Dec(k, c'_i)$ (can repeat poly-times)
4. A sends two messages m^*_0, m^*_1
5. C replies with a ciphertext $c^* \leftarrow\$ Enc(k, m^*_b)$ $(b \in \{0,1\})$
6. A sends a message m_i to C and C sends back a ciphertext $c_i \leftarrow\$ Enc(k, m_i)$ (can repeat poly-times)
7. A sends a cipher c'_i to C and C sends back the plaintext $m'_i = Dec(k, c'_i)$ (can repeat poly-times)
8. A sends b'

A wins if $b' = b$; i.e. A understood whether C encrypted m_0 or m_1.
Notice that 2. $\equiv$ 6. and 3. $\equiv$ 7.

3.11 Authenticity

It should be hard to produce a **valid** cipher without knowing the key, so the decryption algorithm can also output a symbol $\perp$ if the ciphertext is invalid.

$$Dec : \mathcal{K} \times \mathcal{C} \to \mathcal{M} \cup \{\perp\}$$

$\Pi = (Enc, Dec)$ satisfies authenticity if $\forall$PPT A $Pr[GAME^{AUTH}_{\Pi,A}(\lambda) = 1] \leq negl(\lambda)$

$GAME^{AUTH}_{\Pi,A}(\lambda)$:

1. C samples a key $k \leftarrow\$ \mathcal{K}$
2. A sends a message m_i to C and C replies with $c_i \leftarrow\$ Enc(k, m_i)$ (can repeat poly-times)
3. A sends a cipher c^*

Output 1 if both of the followings are true:

- $Dec(k, c^*) \neq \perp$
- $c^* \notin \{c_i\}$

Number Theory

4.1 Modular arithmetic

Consider $\mathbb{Z}_n = \{0, 1, ..., n-1\}$.
On integers *mod n*, $(\mathbb{Z}_n, +)$ is a group:

- **Identity**: $\exists 0 \in \mathbb{Z}_n$ s.t. $\forall a \in \mathbb{Z}_n \ a + 0 = a \ mod \ n$
- **Addition**: $\forall a, b \in \mathbb{Z}_n \ a + b = b + a \ mod \ n$
- **Inverse**: $\forall a \in \mathbb{Z}_n \ \exists a^{-1} \in \mathbb{Z}_n$ s.t. $a + a^{-1} = 0$

Can be proven that $(\mathbb{Z}_n, \cdot)$ is not always a group.

4.2 GroupGen

Let $GroupGen(1^\lambda)$ be an algorithm that takes as input a security parameter λ and returns the description of a group.

$$(\mathbb{G}, g, q) \leftarrow\$ \ GroupGen(1^\lambda)$$

$\mathbb{G}$ is a **cyclic group**
g is the **generator** of the group
q is the **order** of the group

4.3 Discrete Log - DL

$\mathbb{G}$ is usually $\mathbb{Z}_p^*$, where p is a large prime.
Note that because p is a prime then $q = p - 1$.

GAME:

1. C samples $(\mathbb{G}, g, q) \leftarrow\$ \ GroupGen(1^\lambda)$
2. C samples $x \leftarrow\$ \ \mathbb{Z}_p$

   my notes say $\mathbb{Z}_q$; I think it's wrong

3. C calculates $y = g^x$ in $\mathbb{G}$ (i.e. $g^x \ mod \ p$)
4. C sends $(\mathbb{G}, g, q)$ and y

5. A sends back x'

A wins if $y = g^{x'}$.

Assumption: DL defines a OWF.

4.4 Decisional Diffie Hellman - DDH

Consider the following games.

REAL:

1. C samples $(\mathbb{G}, g, q) \leftarrow\$ \, GroupGen(1^\lambda)$
2. C samples $x, y \leftarrow\$ \, \mathbb{Z}_n$
3. C calculates $X = g^x$, $Y = g^y$, $Z = g^{xy}$
4. C sends $(\mathbb{G}, g, q), X, Y, Z$ to A
5. A sends b'

RAND:

1. C samples $(\mathbb{G}, g, q) \leftarrow\$ \, GroupGen(1^\lambda)$
2. C samples $x, y, z \leftarrow\$ \, \mathbb{Z}_n$
3. C calculates $X = g^x$, $Y = g^y$, $Z = g^z$
4. C sends $(\mathbb{G}, g, q), X, Y, Z$ to A
5. A sends b'

b' says whether A thinks Z is g^{xy} or g^z.
A wins if it understands where Z comes from.

Public Key Encryption

5.1 Key Generation

Each party can have a **public key** and a **secret key**. These two keys will be randomly sampled from an algorithm that generates them. This means that in PKE a primitive will also be composed of the key generation algorithm, besides the encryption and decryption ones.

$$\Pi = (KGen, Enc, Dec)$$

A party will samples key in the following way:

$$(p_k, s_k) \leftarrow\$ \ KGen(1^\lambda)$$

5.2 TrapDoor Permutation - TDP

The idea is that a party has a secret key; without the key the permutation is like a OWF, it can't be inverted (unless with negligible probability).

$\Pi = (KGen, f_{pk}, f_{sk}^{-1})$
$(p_k, s_k) \leftarrow\$ \ KGen(1^\lambda)$
$f_{pk} : X_{pk} \to X_{pk}$
$f_{sk} : X_{pk} \to X_{pk}$
By $KGen$: $\forall x \in X_{pk} \ \forall (pk, sk) \ f_{sk}^{-1}(f_{pk}(x)) = x$

Consider the following game:

1. C samples $(p_k, s_k) \leftarrow\$ \ KGen(1^\lambda)$
2. C samples $x \leftarrow\$ \ X_{pk}$
3. C calculates $y = f_{pk}(x)$
4. C sends y to A
5. A sends back x'

A wins if $x' = x$

5.3 RSA

Define $n = p \cdot q$, where p and q are two primes. $X_{pk} = \mathbb{Z}_n$.
Fix some e. $d = e^{-1}\ mod\ \varphi(n) = e^{-1}\ mod\ (p-1)(q-1)$

$f_{pk}(x) = x^e\ mod\ n = c$
$f_{sk}^{-1}(c) = y^d\ mod\ n$

Consider p, q, n, e, d as defined above, and consider the following game:

1. C samples $x \leftarrow\$ X_{pk}$
2. C calculates $y = x^e\ mod\ n$
3. C sends $(n, e), y$ to A
4. A sends back x'

A wins if $x' = x$

5.4 Hash Proof Systems

Both Alice and Bob have a problem y, but Alice also knows a solution x for y. She wants to convince Bob that she know the solution to the problem, possibly without revealing x.
$y \in L$, where $L \subseteq NP$. x is s.t. $R(x, y) = 1$.

y is a **statement**.
x is a **valid witness** for y.

There is a $Setup(1^\lambda)$ algorithm ran by a trusted part.
There are also two new algorithms: $Prove$ and $Verify$.

The interation is the following:

1. Trusted part runs $(\omega, \tau) \leftarrow\$ Setup(1^\lambda)$
2. Alice knows x, y, ω
3. Bob knows y, τ
4. Alice samples $\pi \leftarrow\$ Prove(\omega, y, x)$
5. Alice sends π to Bob
6. Bob calculates $\tilde{\pi} = Verify(\tau, \pi)$

$y \in L \leftrightarrow \tilde{\pi} = \pi$

5.5 t-universality

$\Pi = (Setup, Prove, Verify)$ is t-universal if $\forall$ distinct statements $y_1, ..., y_t$ s.t. $\forall i \in [t]$ $y_i \notin L$, the following holds:

$$(\omega, Ver(\tau, y_1), ..., Ver(\tau, y_t)) \equiv (\omega, v_1, ..., v_t)$$

Where:

- $(\omega, \tau) \leftarrow\$ \; Setup(1^\lambda)$
- $v_1, ..., v_t \leftarrow\$ \; \mathcal{P}$

$\mathcal{P}$ is the space where proves $(\pi, \tilde{\pi})$ are defined.

5.6 Membership indistinguishability - MI

The idea is that a language is membership indistinguishable if it's hard to understand whether a statement is in the language or not.

A language $L \subseteq NP$ is MI if $\exists \bar{L}$ s.t.

- $L \cap \bar{L} = \varnothing$ and $L \cup \bar{L} = \mathcal{Y}$
- $\exists$ algorithm $Samp$ that generates x, y s.t.
  - $y \leftarrow\$ \; \mathcal{Y}$, $y \in L$
  - $R(x, y) = 1$
- $\exists$ algorithm $\overline{Samp}$ that generates $y \leftarrow\$ \; \mathcal{Y}$ s.t. $y \in \bar{L}$
- $\{y | (y, x) \leftarrow\$ \; Samp(1^\lambda)\} \approx_c \{y | y \leftarrow\$ \; \overline{Samp}(1^\lambda)\}$

Digital signatures

We will still use $KGen(1^\lambda)$, but also two new algorithms: $Sign$ and $Verify$.

Signature

$$\sigma \leftarrow\!\$\ Sign(m, sk)$$

m is a message and sk a secret key sampled from $KGen$. σ is a signature for m.

Verification

$$Verify((m, \sigma), pk) = \{0, 1\}$$

The output says whether σ is a valid signature for m.

6.1 UF-CMA

$\Pi = (Kgen, Sign, Verify)$ is UF-CMA if $\forall$PPT A $Pr[GAME_{\Pi,A}^{UF-CMA}(\lambda) = 1] \leq negl(\lambda)$

$GAME_{\Pi,A}^{UF-CMA}(\lambda)$:

1. C samples $(p_k, s_k) \leftarrow\!\$\ KGen(1^\lambda)$
2. A sends message m to C, C calculates $\sigma \leftarrow\!\$\ Sign(sk, m)$ and sends it back to A (can repeat poly-times)
3. A sends m^*, σ^* to C

Output 1 if $Verify(pk, m^*, \sigma^*) = 1$

6.2 Transcript

Alice not only wants to convince Bob that she has Bob's pk $\boxed{\text{is that right?}}$ but she also wants to convince him that she's indeed Alice.

We now have an algorithm $\mathcal{P}$, for the Prover Alice, and an algorithm $\mathcal{V}$, for the verifier Bob.

$$\tau \leftarrow\!\$\ (\mathcal{P}(pk, sk) \rightleftarrows \mathcal{V}(pk))$$

This construction grants Passive security.

6.3 Passive security

$\Pi = (Kgen, \mathcal{P}, \mathcal{V})$ is passive-secure if $\forall$PPT A $Pr[GAME^{ID}_{\Pi,A}(\lambda) = 1] \leq negl(\lambda)$

$GAME^{ID}_{\Pi,A}(\lambda)$:

1. C samples $(p_k, s_k) \leftarrow\$ \ KGen(1^\lambda)$
2. C sends pk to A
3. A sends an empty query to C and C replies with $\tau \leftarrow\$ \ (\mathcal{P}(pk, sk) \rightleftarrows \mathcal{V}(pk))$ (can repeat poly-times)
4. Run many interactions on "Impersonation"

Return 1 if at the end of the "Impersonation" part $\mathcal{V}$ returns 1

ID schemes that grant passive security can be built with:

- Honest Verifier Zero Knowledge - HVZK
- Special Soundness - SS

6.4 Honest Verifier Zero Knowledge - HVZK

$\Pi = (KGen, \mathcal{P}, \mathcal{V})$ is HVZK if $\exists$PPT Sim (Simulator) s.t.

$$REAL^{ID}_{\Pi}(\lambda) \approx_c SIMU^{ID}_{\Pi}(\lambda)$$

$REAL^{ID}_{\Pi}(\lambda) \equiv \{pk, sk, (\mathcal{P}(pk, sk) \rightleftarrows \mathcal{V}(pk))|(p_k, s_k) \leftarrow\$ \ KGen(1^\lambda)\}$

$SIMU^{ID}_{\Pi}(\lambda) \equiv \{pk, sk, Sim(pk)|(p_k, s_k) \leftarrow\$ \ KGen(1^\lambda)\}$

The idea is that honestly computed transcripts reveal nothing on sk, as it can also be obtained by running $Sim(pk)$

6.5 Special Soundness - SS

Let Π be a $\Sigma{-}protocol$ $\boxed{\text{Explanation of } \Sigma - protocol \text{ has been skipped for now}}$.
Π is SS if $Pr[GAME^{SS}_{\Pi,A})(\lambda) = 1] \neq negl(\lambda)$

$GAME^{SS}_{\Pi,A})(\lambda)$:

1. C samples $(p_k, s_k) \leftarrow\$ \ KGen(1^\lambda)$
2. C sends pk to A
3. A sends $\alpha, \beta, \beta', \gamma, \gamma'$

Output 1 if $\beta \neq \beta'$ and $\mathcal{V}(pk, (\alpha, \beta, \gamma)) = \mathcal{V}(pk, (\alpha, \beta', \gamma')) = 1$.

Signatures without ROM

7.1 Bilinear Groups

Let $BilGroupGen(1^\lambda)$ be an algorithm that returns the following description:

$$(\mathbb{G}, \mathbb{G}_T, q, g, \hat{e}) \leftarrow\$ \; BilGroupGen(1^\lambda)$$

- $\mathbb{G}$: cyclic group (moreover an elliptic curve group)
- $\mathbb{G}_T$: target group
- q: order of both Groups
- g: generator of $\mathbb{G}$
- $\hat{e}$: **pairing** function

$\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ s.t.

- $\forall g, h \in \mathbb{G} \; \forall a, b \in \mathbb{Z}_q \;\; \hat{e}(g^a, h^b) = \hat{e}(g, h)^{ab}$
- $\hat{e}(g, g) \neq 1$ (non degenerate)

7.2 Construction of signatures

Let $\Pi = (KGen, Sign, Verify)$

$\boldsymbol{KGen(1^\lambda)}$:
$params : (\mathbb{G}, \mathbb{G}_T, q, g, \hat{e}) \leftarrow\$ \; BilGroupGen(1^\lambda)$
$a \leftarrow\$ \; \mathbb{Z}_q$
$g_1 = g^a$
$g_2, \mu_0, ..., \mu_k \leftarrow\$ \; \mathbb{G}$ ($\mu_0, ..., \mu_k$ correspond to some kind of "hash function")

$pk = (params, g_1, g_2, \mu_0, ..., \mu_k)$
$sk = g_2^a$

$\boldsymbol{Sign(sk, m)}$:
$m = (m[1], ..., m[k]) \in \{0, 1\}^k$
$\alpha_{\mu 0, ..., \mu_k}(m) = \mu_0 \cdot \sum_{i=1}^{k} \mu_i^{m[i]}$

$r \leftarrow\$ \; \mathbb{Z}_q$
$\sigma = (sk \cdot \alpha(m)^r, g^r) = (g_2^a \cdot \alpha(m)^r, g^r) = (\sigma_1, \sigma_2)$

Verify(pk, m, σ)

Check $\hat{e}(g, \sigma_1) = \hat{e}(\sigma_2, \alpha(m)) \cdot \hat{e}(g_1, g_2)$