Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Tighter Security for Group Key Agreement in the Random Oracle Model

Bachelor Thesis

A. Ellison

January 19, 2038

Advisors: Prof. Dr. D. Hofheinz, Dr. K. Klein

Department of Computer Science, ETH Zürich

Abstract

TODO: How to adapt abstract? What should it contain?

The Messaging Layer Security (MLS) protocol, recently standardized in RFC 9420 [2], aims to provide efficient asynchronous group key establishment with strong security guarantees. TreeKEM is the construction underlying MLS and a variant of it was proven adaptively secure in the Ran- dom Oracle Model (ROM) with a polynomial loss in security in [1]. The proof makes use of the Generalized Selective Decryption (GSD) security game introduced in [6], adapted to the public-key setting. GSD security is closely related to the security of TreeKEM and the encryption scheme used in TreeKEM was proven to be GSD secure in the ROM under the standard assumption of IND-CPA security, implying a proof of security for TreeKEM (a sketch of this proof was provided in [1] for the TreeKEM variant).

TODO: describe results

Contents

Chapter 1

Introduction

TODO: more accessible introduction on why this is important We all rely on messaging applications like WhatsApp, Signal, etc. in our daily lives and take it for granted that our messages will be transmitted securely (**TODO:** "see it as a prerequisite" maybe better?). **TODO:** smoother transition to talking about protocols? For two parties, the Double Ratchet protocol is a common solution (**TODO:** true?) to transmit messages securely and efficiently. For more than two parties this problem was only solved recently with the MLS protocol.

The Messaging Layer Security (MLS) protocol, recently standardized in RFC 9420 [2], aims to provide efficient asynchronous group key establishment with strong security guarantees. The main component of MLS, which is the source of its important efficiency and security properties, is a protocol called TreeKEM (initially proposed in [3]). In essence, TreeKEM, as adopted from its predecessors, structures a group of users as a binary tree with the group key at the root and all group members as leaves. Group members may then compute the group key, update it or add/remove other members with a number of operations logarithmic in the group size.

As for any scheme, it is important to have formal security guarantees for TreeKEM based on precise hardness assumptions. Providing security definitions for the scheme already helps to describe exactly what assumptions are made on the capabilities of an adversary and what kind of security one should expect when using the scheme in practice. Moreover, proofs of (reasonably tight) security under these definitions serve as a guide to implementors on what values to choose for the security parameters of the scheme and provide strong justification that there are no flaws in its design. Given that a major vision for the MLS protocol is for it to be used by messaging applications and that it has support from several large companies ([4], [5]), it has the potential to be used by a huge number of users. Thus, it is important to better understand the security of MLS and hence also of TreeKEM.

One choice that can be made when defining the security of TreeKEM is whether the adversary is modeled as *selective* or *adaptive*. In the former case, the adversary must provide all the interactions it will have with the protocol and when it will attempt to break the scheme at the beginning of the security game, while in the latter case the adversary can make its decisions based on responses from previous interactions. Clearly, the adaptive setting is much closer to how an attack would unfold in practice, so it is desirable to prove security against adaptive adversaries. However, achieving this without too much of a blow-up in the security loss is a challenge since one often resorts to guessing actions performed by the adversary.

The Generalized Selective Decryption (GSD) security game ([6]) was introduced precisely to analyze adaptive security for protocols based on a graph-like structure (as is the case with TreeKEM). In [1], a variant of TreeKEM was proven adaptively secure in the Random Oracle Model (ROM) with a security loss in $\mathcal{O}((n \cdot Q)^2)$, where n is the number of users and Q the number of protocol operations performed by these users. The proof mainly relies on showing that the encryption scheme employed in TreeKEM, a slight modification of an arbitrary IND-CPA secure encryption scheme, is GSD secure in the ROM.

TODO: describe results and contribution in detail

Chapter 2

Preliminaries

TODO: Define private-key encryption scheme.

TODO: Define public-key encryption scheme.

Definition 2.1 (The IND-CPA Game) *Let Π a private-key encryption scheme. Define the game $\mathrm{Game}^{\mathrm{IND-CPA}}_{\mathcal{A},\Pi}$ for an adversary $\mathcal{A}$:*

1. *A key $k \leftarrow \mathrm{Gen}()$ is generated.*
2. *The adversary $\mathcal{A}$ is given oracle access to $\Pi.\mathrm{Enc}_k$ and outputs a pair of messages m_0, m_1 of the same length.*
3. *A bit $b \leftarrow \{0,1\}$ is sampled and $\mathcal{A}$ is given the ciphertext $c \leftarrow \mathrm{Enc}_k(m_b)$. ($\mathcal{A}$ continues to have oracle access to $\Pi.\mathrm{Enc}_k$.)*
4. *$\mathcal{A}$ outputs a bit b'. The output of the game is defined to be 1 if $b' = b$, and 0 otherwise.*

Definition 2.2 (IND-CPA security) *A private-key encryption scheme Π is (t, ε, q)-*IND-CPA secure *if for any adversary $\mathcal{A}$ running in time t we have*

$$\mathrm{Adv}^{\mathrm{MI-EAV}}_{\Pi}(\mathcal{A}) := 2 \cdot \left| \Pr\left[\mathrm{Game}^{\mathrm{MI-EAV}}_{\mathcal{A},\Pi} = 1\right] - \frac{1}{2} \right| \leq \varepsilon.$$

TODO: Shortly motivate EAV security and reference Katz and Lindell.

Definition 2.3 (The EAV Game) *Let Π a private-key encryption scheme. Define the game $\mathrm{Game}^{\mathrm{EAV}}_{\mathcal{A},\Pi}$ for an adversary $\mathcal{A}$:*

1. *A key $k \leftarrow \mathrm{Gen}()$ is generated.*
2. *The adversary $\mathcal{A}$ outputs a pair of messages m_0, m_1 of the same length.*
3. *A bit $b \leftarrow \{0,1\}$ is sampled and $\mathcal{A}$ is given the ciphertext $c \leftarrow \mathrm{Enc}_k(m_b)$.*

4. *$\mathcal{A}$ outputs a bit b'. The output of the game is defined to be 1 if $b' = b$, and 0 otherwise.*

Definition 2.4 (EAV security) *A private-key encryption scheme Π is (t, ε)-*EAV secure *if for any adversary $\mathcal{A}$ running in time t we have*

$$\mathrm{Adv}_{\Pi}^{\mathrm{EAV}}(\mathcal{A}) := 2 \cdot \left| \Pr\left[\mathrm{Game}_{\mathcal{A},\Pi}^{\mathrm{EAV}} = 1\right] - \frac{1}{2} \right| \leq \varepsilon.$$

Lemma 2.5 *Let Π a private-key encryption scheme. If Π is (t, ε)-IND-CPA secure, then Π is (t, ε)-EAV secure.*

Proof This follows immediately from the fact that any EAV adversary is also an IND-CPA adversary. □

TODO: explain the ROM

Chapter 3

Tighter GSD security

TODO: Motivate GSD

Following the general approach used in [1] to prove the security of (a variant of) TreeKEM in the ROM, we first prove a result on the GSD security of an IND-CPA secure encryption scheme. We do this specifically for the DHIES scheme. Moreover, we will make some notable modifications to the public-key GSD game defined in [1], to allow for results to be applied to TreeKEM more directly. We motivate the modifications made later in Section 4 on page 17.

3.1 Seeded GSD with Dependencies

We call our adaptation of GSD security *Seeded GSD with Dependencies* (SD-GSD).

TODO: Explain definition in words. **TODO:** Motivate restrictions to the adversary. **TODO:** Do not allow cycles in $(V, E \cup D)$ either. **TODO:** Add remark that cycles are (maybe) ok in the ROM.

Definition 3.1 (The SD-GSD Game) *Let $\lambda \in \mathbb{N}$ a security parameter.* ***Q:*** *Where to define λ? Let $\Pi = (\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec})$ a public-key encryption scheme. Let $H_{\mathrm{gen}}, H_{\mathrm{dep}} \colon \{0,1\}^\lambda \to \{0,1\}^\lambda$ two KDFs. Define the game $\mathrm{Game}^{\mathrm{SD-GSD}}_{\mathcal{A},\Pi}$ for an adversary $\mathcal{A}$:*

1. *The adversary $\mathcal{A}$ outputs $n \in \mathbb{N}$ and a list of dependencies $D = \{(a_i, b_i)\}_{i=1}^m \in [n]^2$. For each $v \in [n]$:*

   (i) • ***Case $v = b_i$ for some i (v is the target of some dependency):*** *set $s_v = H_{\mathrm{dep}}(s_{a_i})$.*

   • ***Otherwise:*** *sample $s_v \leftarrow \{0,1\}^\lambda$.*

   We call s_v the seed *of the node v and a tuple $(a, b) \in D$ a* seed dependency.

(ii) *Compute* $(sk_v, pk_v) = \mathsf{Gen}(H_{\text{gen}}(s_v))$. ***TODO:*** *Define what RHS means.*

Set $\mathcal{C} = E = \varnothing$. *We call the directed graph* $([n], E)$ *a* GSD graph *of* size n.

2. $\mathcal{A}$ *may adaptively do the following queries:*
   - $\mathsf{reveal}(v)$ *for* $v \in [n]$: $\mathcal{A}$ *is given* pk_v.
   - $\mathsf{encrypt}(u, v)$ *for* $u, v \in [n], u \neq v$: (u, v) *is added to* E *and* $\mathcal{A}$ *is given* $c \leftarrow \mathsf{Enc}_{pk_u}(s_v)$.
   - $\mathsf{corrupt}(v)$ *for* $v \in [n]$: $\mathcal{A}$ *is given* s_v *and* v *is added to* $\mathcal{C}$. *We call such a node* $v \in \mathcal{C}$ corrupted. *All nodes not reachable from any corrupted node in the graph* $([n], E \cup D)$ *are* safe *(while all other nodes are* unsafe*) and we say their seeds are* hidden *(even if an unsafe node happens to have the same seed).*
3. $\mathcal{A}$ *outputs a node* $v \in [n]$. *We call* v *the* challenge node. *A bit* $b \leftarrow \{0,1\}$ *is sampled and* $\mathcal{A}$ *is given*

$$r = \begin{cases} H_{\text{dep}}(s_v) & b = 0 \\ s & b = 1 \end{cases},$$

   where $s \leftarrow \{0,1\}^{\lambda}$. $\mathcal{A}$ *may continue to do queries as before.*
4. $\mathcal{A}$ *outputs a bit* b'. *The output of the game is defined to be* 1 *if* $b' = b$, *and* 0 *otherwise.*

We require an adversary playing the above game to adhere to the following:

- *The challenge node always remains a sink.*
- *The challenge node is safe.*
- reveal *is never queried on the challenge node.*
- *The graphs* (V, E) *and* (V, D) *always remain acyclic and without self-loops.*
- *All paths in the graph* (V, D) *are vertex disjoint.*

TODO: Remove random oracles from SD-GSD security and add them to Theorem 3.3 on the next page instead.

Since we are only interested in the security of the SD-GSD game for the case where H_{gen} and H_{dep} are random oracles, we directly assume in our definition that the KDFs are modelled as such.

Definition 3.2 (SD-GSD security in the ROM) *A public-key encryption scheme* Π *is* $(t, \varepsilon, N, \delta)$-SD-GSD secure *if for any adversary* $\mathcal{A}$ *constructing a GSD graph of size at most* N *and indegree at most* δ *and running in* t *time we have*

$$\mathsf{Adv}_{\Pi}^{\mathsf{SD-GSD}}(\mathcal{A}) := 2 \cdot \left| \Pr\left[\mathsf{Game}_{\mathcal{A},\Pi}^{\mathsf{SD-GSD}} = 1\right] - \frac{1}{2} \right| \leq \varepsilon$$

when H_{gen} *and* H_{dep} *are random oracles.*

3.2 Proving SD-GSD security for DHIES

TODO: Add assumption that Π_s.Gen samples uniformly from $\{0,1\}^x$ **TODO:** Comment on switch from IND-CPA security to EAV security.

Theorem 3.3 *Let $N, \delta \in \mathbb{N}$ arbitrary with $\delta \leq N$. Let Π_{DH} the DHIES scheme instantiated with a private-key encryption scheme Π_s where Π_s.Gen samples a key uniformly at random from $\{0,1\}^\lambda$. Let H_{DH} the KDF and $\mathbb{G}$ the group used in Π_{DH}. If Π_s is (t, ε)-EAV secure, the DDH problem is (t, ε)-hard in $\mathbb{G}$ and H_{DH} is modelled as a random oracle, then Π_{DH} is $(\tilde{t}, \tilde{\varepsilon}, N, \delta)$-SD-GSD secure with*

$$\begin{aligned}\tilde{t} = t - \mathcal{O}\big(m_s + & m_{\text{DH}} \\ & + N \cdot (t_{H_{\text{dep}}} + t_{\text{sample}} + t_{H_{\text{gen}}} + t_{\Pi_{\text{DH}}.\text{Gen}}) \\ & + N^2 \cdot t_{\Pi_{\text{DH}}.\text{Enc}}\big).\end{aligned}$$

and

$$\tilde{\varepsilon} = \dots$$

and where m_s is an upper bound on the number of queries made to either H_{gen} or H_{dep} and m_{DH} is an upper bound on the number of queries made to m_{DH}. ***TODO:*** *Nicer way to introduce m_s and m_{DH}?*

Intuition Consider an arbitrary SD-GSD adversary $\mathcal{A}$. For an execution of $\text{Game}^{\text{SD-GSD}}_{\mathcal{A},\Pi_{\text{DH}}}$ we say "*$\mathcal{A}$ wins*" to denote the event $\text{Game}^{\text{SD-GSD}}_{\mathcal{A},\Pi_{\text{DH}}} = 1$. As usual with random oracles we proceed by a case distinction on whether they were queried on some interesting value. Accordingly, let Q_x denote the event that $\mathcal{A}$ queries H_x on a hidden seed for $x \in \{\text{gen}, \text{dep}\}$. (**Q:** What if corrupted seed is queried and it happens to coincide with a hidden seed?) Then we can write

$$\begin{aligned}\Pr[\mathcal{A} \text{ wins}] &= \Pr[\mathcal{A} \text{ wins} \wedge Q_{\text{dep}}] + \Pr[\mathcal{A} \text{ wins} \wedge \overline{Q_{\text{dep}}}] \\ &\overset{(*)}{=} \Pr[\mathcal{A} \text{ wins} \wedge Q_{\text{dep}}] + \frac{1}{2} \\ &\leq \Pr[Q_{\text{dep}}] + \frac{1}{2} \\ &\leq \Pr[Q_s] + \frac{1}{2},\end{aligned}$$

where $Q_s := Q_{\text{gen}} \cup Q_{\text{dep}}$ (s for *seed*).

TODO: Justify (*). (And perhaps name it better?)

TODO: Motivate why we introduce Q_s. (Reason: If we try to bound Q_{dep} by itself, we must separately deal with the case where the adversary was able to trigger it at a node v by triggering Q_{gen} at a parent node p and subsequently decrypting a ciphertext. But our argument To eliminate this, we want to look at the point in time where either of the two events was first triggered.)

The heart of the proof is to bound $\Pr[Q_s]$. When the adversary first triggers Q_s by querying the seed of some hidden node w, it can only have learned the seed through encryptions $c_1 \leftarrow \Pi_{\text{DH}}.\text{Enc}_{pk_{u_1}}(s_w), \dots, c_d \leftarrow \Pi_{\text{DH}}.\text{Enc}_{pk_{u_d}}(s_w)$ where $(u_1, w), \dots, (u_d, w)$ are edges in the GSD graph (obtained through corresponding queries $\text{encrypt}(u_1, w), \dots, \text{encrypt}(u_d, w)$).

TODO: Add plot illustrating edges in GSD graph and a potential seed dependency. Add note why no information is learned through H_{DH}. (No information is learned since all parent nodes of w are also safe and querying $H_{\text{DH}}(s_p)$ for a parent p would thus trigger Q_{dep}.)

The proof in [1] simply argued that this is not too likely if these encryptions were made with an IND-CPA secure scheme. In the context of the DHIES scheme we can say more about these encryptions and achieve a better reduction loss. Let $x_i = \log_g(pk_{u_i})$. Each encryption c_i is a tuple of the form $\langle g^{y_i}, \Pi_s.\text{Enc}_{k_i}(s_w)\rangle$ where $y_i \leftarrow [|\mathbb{G}|], k_i = H_{\text{DH}}(g^{x_i \cdot y_i})$. Now we can again do a case distinction on whether H_{DH} was queried for some group element $g^{x_j \cdot y_j}$ or not.

- If such a query was made, then $\mathcal{A}$ solved the Diffie-Hellman challenge (g^{x_j}, g^{y_j}). (Remember that we assumed that w is the first node for which Q_s is triggered and if the seed of w is hidden, then so are the seeds of the nodes u_i. Thus the adversary has not learned the exponent x_i through querying $H_{\text{gen}}(s_{u_i})$ for any i.)
- If no such query was made, then from $\mathcal{A}$'s perspective all the k_i are independent, uniformly random keys and it still was able to learn s_w from the EAV secure encryptions $\Pi_s.\text{Enc}_{k_1}(s_w), \dots, \Pi_s.\text{Enc}_{k_d}(s_w)$.

We can bound the probability of either of these events occurring using hardness of the DDH problem in $\mathbb{G}$ and EAV security of Π_s, respectively.

To this end, we call a group element $k \in \mathbb{G}$ a *hidden Diffie-Hellman key* if $k = pk_u^{y_{u,w}}$, where (u, w) is an edge in the GSD graph, u is safe and $y_{u,w}$ is the exponent chosen in the DHIES encryption of s_w (i.e. $\mathcal{A}$ was given a ciphertext of the form $\langle g^{y_{u,w}}, \dots\rangle$ when it queried $\text{encrypt}(u, w)$). Now analogously to above let Q_{DH} the event that $\mathcal{A}$ queries H_{DH} on a hidden Diffie-Hellman key and let F_{DH} the event that $\mathcal{A}$ triggers Q_{DH} *before* having triggered Q_s. Then we can split of the event Q_s into two cases:

$$\Pr[Q_s] = \Pr[Q_s \wedge F_{\text{DH}}] + \Pr\left[Q_s \wedge \overline{F_{\text{DH}}}\right].$$

We bound $\Pr[Q_s \wedge F_{\text{DH}}]$ and $\Pr[Q_s \wedge F_{\text{DH}}]$ in Lemma 3.7 and Lemma 3.8, respectively, which overall gives us a bound on the advantage of $\mathcal{A}$ using (3.2).

Proof (of Theorem 3.3) Let $\mathcal{A}$ an arbitrary SD-GSD adversary running in time $\tilde{t}$. We will use the events defined above. We first justify step $(*)$ in (3.2).

By Lemma 3.4 we have

$$\Pr[Q_s \wedge F_{DH}] \leq \dots$$

and by Lemma 3.8 on page 11 we have

$$\Pr\left[Q_s \wedge \overline{F_{DH}}\right] \leq \dots$$

Then by (3.2)

$$\Pr[\mathcal{A} \text{ wins}] \leq x + \frac{1}{2},$$

so

$$\mathrm{Adv}_{\Pi}^{\mathrm{SD-GSD}}(\mathcal{A}) \leq 2 \cdot |x| = \tilde{\varepsilon}. \qquad \square$$

3.2.1 Reducing to the DDH problem

Lemma 3.4 *Let $\mathcal{A}$ an SD-GSD adversary. Let $\Pi_{DH}, H_{DH}, \mathbb{G}$ and the events Q_s, Q_{DH}, F_{DH} as in the statement and proof of Theorem 3.3 on page 7 and assume that the DDH problem is (t, ε)-hard in $\mathbb{G}$. Then*

$$\Pr[Q_s \wedge F_{DH}] \leq \dots.$$

Proof **TODO:** Make a note that we only care about Q_{DH} being triggered before Q_{gen} for the proof, but we need the remaining information about Q_{dep} in Lemma 3.8 on page 11 $\square$

3.2.2 Reducing to EAV security

TODO: Motivation for MI-EAV

Definition 3.5 (The MI-EAV Game) *Let Π a private-key encryption scheme. Define the game $\mathrm{Game}_{\mathcal{A},\Pi}^{\mathrm{MI-EAV}}$ for an adversary $\mathcal{A}$:*

1. *The adversary $\mathcal{A}$ outputs $q \in \mathbb{N}$ and a pair of messages m_0, m_1 of the same length. We refer to q as the number of* queries *made by $\mathcal{A}$.*
2. *A bit $b \leftarrow \{0, 1\}$ is sampled. For each $i \in [q]$, $\mathcal{A}$ is given an encryption $c_i \leftarrow \Pi.\mathrm{Enc}_{k_i}(m_b)$ where $k_i \leftarrow \Pi.\mathrm{Gen}()$ is generated independently from the other keys.*
3. *$\mathcal{A}$ outputs a bit b'. The output of the game is defined to be 1 if $b' = b$, and 0 otherwise.*

Definition 3.6 (MI-EAV security) *A private-key encryption scheme Π is (t, ε, q)-*MI-EAV secure *if for any adversary $\mathcal{A}$ making at most q queries and running in time t we have*

$$\mathrm{Adv}_{\Pi}^{\mathrm{MI-EAV}}(\mathcal{A}) := 2 \cdot \left| \Pr\left[\mathrm{Game}_{\mathcal{A},\Pi}^{\mathrm{MI-EAV}} = 1\right] - \frac{1}{2} \right| \leq \varepsilon.$$

Similar to how IND-CPA security for a single encryption query implies IND-CPA security for q queries with a security loss of q by a standard hybrid argument, we can show that EAV security implies MI-EAV security with the same loss. Given the well known result for IND-CPA security it seems intuitive that one should be able to adapt the hybrid argument to show MI-EAV security from IND-CPA security. To see why we can make do with EAV security, recall the hybrid argument for IND-CPA security: We define the sequence of hybrid games $H_0, \ldots, H_q$ where in the game H_i always the second message is encrypted for the first i encryption queries and always the first for the remaining $q - i$ queries. Then given an IND-CPA adversary $\mathcal{A}$ for multiple encryptions, an IND-CPA adversary $\mathcal{A}'$ is constructed to bound

$$|\Pr[\mathcal{A} \text{ outputs 1 in game } H_{i-1}] - \Pr[\mathcal{A} \text{ outputs 1 in game } H_i]|$$

for arbitrary i. When $\mathcal{A}'$ simulates H_{i-1} or H_i to $\mathcal{A}$ depending on which message from the i-th query gets encrypted by the IND-CPA challenger, it makes use of the encryption oracle in the IND-CPA security game to pass on the right encryptions to $\mathcal{A}$ for all other queries. Now notice that if we wanted to simulate to an MI-EAV adversary we wouldn't need access to an encryption oracle since for the MI-EAV security game all the other encryptions can easily be generated by manually sampling the new keys.

Lemma 3.7 *Let Π a private-key encryption scheme with finite message space. Let $t_{\text{Gen}}, t_{\text{Enc}}$ upper bounds for the runtime of Π.Gen and Π.Enc, respectively. If Π is (t, ε)-EAV secure, then for all $q \in \mathbb{N}$, Π is $(\tilde{t}, q \cdot \varepsilon, q)$-MI-EAV secure with $\tilde{t} = t - \mathcal{O}(q \cdot (t_{\text{Gen}} + t_{\text{Enc}}))$.*

Proof Note that since the message space is finite, the time to encrypt a message is bounded. As outlined above the Lemma follows from a simple hybrid argument. Let $q \in \mathbb{N}$ and $\mathcal{A}$ an arbitrary MI-EAV adversary running in time $\tilde{t}$ and making at most q queries. Define the sequence of hybrid games $H_0, \ldots, H_q$ where in the game H_i the first i encryptions given to the adversary encrypt m_1 and all remaining encryptions encrypt m_0. We will write

$$\Pr[\mathcal{A} \to 1 \mid H_i]$$

for the probability of $\mathcal{A}$ outputting 1 when playing the hybrid game H_i.

Let $i \in [q]$. Construct an EAV adversary $\mathcal{A}'$ that behaves as follows:

1. $\mathcal{A}'$ runs $\mathcal{A}$ and gets q, m_0, m_1.
2. $\mathcal{A}'$ outputs the messages m_0, m_1 and gets a ciphertext c from the challenger.
3. $\mathcal{A}'$ gives the ciphertexts $c_1, \ldots, c_q$ to $\mathcal{A}$ where

$$c_j = \begin{cases} \Pi.\text{Enc}_{k_j}(m_1) & i < j \\ c & i = j \\ \Pi.\text{Enc}_{k_j}(m_0) & i > j \end{cases}.$$

and $k_j \leftarrow \Pi.\mathrm{Gen}()\ \forall j$.

4. $\mathcal{A}'$ outputs whatever bit $\mathcal{A}$ outputs.

Now consider the value of the bit b sampled in the EAV game. If $b = 0$, then the first $i-1$ ciphertexts that $\mathcal{A}$ received were encryptions of m_1 and the remaining ciphertexts were encryptions of m_0, where all encryptions were under keys sampled independently with Π.Gen. Thus from the view of $\mathcal{A}$ everything followed the same distribution as in the game H_{i-1} and

$$\Pr\left[\mathcal{A}' \to 1 \mid b = 0\right] = \Pr[\mathcal{A} \to 1 \mid H_{i-1}].$$

Analogously, in the case $b = 1$ the first i ciphertexts received by $\mathcal{A}$ were encryptions of m_1 and the rest encryptions of m_0 so

$$\Pr\left[\mathcal{A}' \to 1 \mid b = 1\right] = \Pr[\mathcal{A} \to 1 \mid H_i].$$

Then

$$\begin{aligned} |\Pr[\mathcal{A} \to 1 \mid H_{i-1}] - {}& \Pr[\mathcal{A} \to 1 \mid H_i]| \\ &= \left|\Pr\left[\mathcal{A}' \to 1 \mid b = 0\right] - \Pr\left[\mathcal{A}' \to 1 \mid b = 1\right]\right| \\ &= \mathrm{Adv}_{\Pi}^{\mathrm{EAV}}(\mathcal{A}') \\ &\le \varepsilon \end{aligned} \tag{3.1}$$

by (t, ε)-EAV security of Π since $\mathcal{A}'$ runs in time $\tilde{t} + \mathcal{O}(q \cdot (t_{\mathrm{Gen}} + t_{\mathrm{Enc}})) = t$. Now let b be the bit sampled in the MI-EAV game. Notice that

$$\Pr[\mathcal{A} \to 1 \mid b = 0] = \Pr[\mathcal{A} \to 1 \mid H_0]$$

and

$$\Pr[\mathcal{A} \to 1 \mid b = 1] = \Pr\left[\mathcal{A} \to 1 \mid H_q\right].$$

Then

$$\begin{aligned} \mathrm{Adv}_{\Pi}^{\mathrm{MI-EAV}}(\mathcal{A}) &= |\Pr[\mathcal{A} \to 1 \mid b = 0] - \Pr[\mathcal{A} \to 1 \mid b = 1]| \\ &= \left|\Pr[\mathcal{A} \to 1 \mid H_0] - \Pr\left[\mathcal{A} \to 1 \mid H_q\right]\right| \\ &= \left|\sum_{i=1}^{q} \Pr[\mathcal{A} \to 1 \mid H_{i-1}] - \Pr[\mathcal{A} \to 1 \mid H_i]\right| \\ &\le \sum_{i=1}^{q} |\Pr[\mathcal{A} \to 1 \mid H_{i-1}] - \Pr[\mathcal{A} \to 1 \mid H_i]| \\ &\overset{(3.1)}{\le} q \cdot \varepsilon. \end{aligned}$$

□

Lemma 3.8 *Recall the variables and events defined in the statement and proof of Theorem 3.3. In particular, assume that Π_{s} is (t, ε)-EAV secure. Let $\mathcal{A}$ an SD-GSD adversary running in time $\tilde{t}$, making at most m_{s} queries to H_{gen} or H_{dep} and at most m_{DH} queries to H_{DH}. Then*

$$\Pr\left[Q_{\mathrm{s}} \wedge \overline{F_{\mathrm{DH}}}\,\right] \le \delta \cdot N \cdot \varepsilon + \frac{m_{\mathrm{s}} \cdot N}{2^{\lambda}}.$$

Intuition By Lemma 3.7 on page 10 we know that Π_s is MI-EAV secure. Continuing the high-level argument before the proof of Theorem 3.3, consider the first moment that $\mathcal{A}$ triggers $Q_s \wedge \overline{F_{\text{DH}}}$ by querying the seed of some safe node w. Since w is safe, a node a that is the source of a seed dependency $(a, w) \in D$ must therefore also be safe. Therefore $\mathcal{A}$ cannot have learned s_w through such a seed dependency by querying H_{dep} because that would have triggered Q_s earlier. **TODO:** Make sure not to repeat previous argument elsewhere. Moreover, as already motivated, it follows from the definition of the event F_{DH} that from $\mathcal{A}$'s perspective all DHIES ciphertexts it got from queries encrypt(u, w) contain encryptions of s_w under independent, uniformly random keys using Π_s. Notice that since H_{dep} was not queried, these ciphertexts were also the only information that $\mathcal{A}$ had at its proposal to learn s_w.

We can use $\mathcal{A}$'s ability to compute the seed s_w of a safe node w from encryptions of s_w to construct an MI-EAV adversary: We first guess the safe node v whose seed $\mathcal{A}$ may query. Next we give the MI-EAV challenger s_w and some other independent seed s, and embed the encryptions we get back into the SD-GSD game when answering queries of the form encrypt(u, w) for any u. If the challenger chooses to encrypt s_w, then $\mathcal{A}$ will trigger the event $Q_s \wedge \overline{F_{\text{DH}}}$, with the same probability as before. If $\mathcal{A}$ triggers $Q_s \wedge \overline{F_{\text{DH}}}$ and we guessed v correctly, we can detect that this happened (by checking if $H_{\text{gen}}(s_w)$ or $H_{\text{dep}}(s_w)$ was queried by $\mathcal{A}$ during the simulation). If the challenger chooses to encrypt s, then $\mathcal{A}$ receives no information about s_w and has negligible probability of querying it. Thus the advantage of the adversary is about $\Pr\left[Q_s \wedge \overline{F_{\text{DH}}}\right]$ and using that Π_s is MI-EAV secure we can bound the probability. Since we are only interested in checking whether the event was triggered for w, the adversary can abort when this is no longer possible (w is corrupted, other hidden seed is queried, etc.).

Proof (of Lemma 3.8) As motivated above we construct an MI-EAV adversary $\mathcal{A}'$ to derive the bound. $\mathcal{A}'$ behaves as follows:

1. $\mathcal{A}'$ runs $\mathcal{A}$ to get n and D and initializes the GSD graph, seeds and set of corrupted nodes as in step 1 of the SD-GSD game.
2. $\mathcal{A}'$ samples $w \leftarrow [n], s \leftarrow \{0,1\}^\lambda$ and gives δ and the messages s_w, s to the challenger. Let $c_1, \dots, c_\delta$ the encryptions it gets back.
3. $\mathcal{A}'$ faithfully simulates the SD-GSD game to $\mathcal{A}$ with the following exception: Whenever $\mathcal{A}$ makes a query of the form encrypt(u, w), $\mathcal{A}'$ replies with $\langle g^x, c_i \rangle$ for $x \leftarrow [|\mathbb{G}|]$ and where i is index of the next ciphertext (from step 2) not yet used.

   During the simulation $\mathcal{A}'$ also pays attention to the following:

   - If any of the following events occur, $\mathcal{A}'$ aborts the simulation and outputs 0:

  - $\mathcal{A}$ queries H_{DH} for a hidden Diffie-Hellman key
  - $\mathcal{A}$ queries H_{gen} or H_{dep} for a hidden seed that is not s_w
  - $\mathcal{A}$ queries $\mathrm{corrupt}(u)$ for some node u such that w is no longer safe

- If $\mathcal{A}$ queries H_{gen} or H_{dep} for s_w, $\mathcal{A}'$ aborts the simulation and outputs 1. This is the only point at which $\mathcal{A}$ outputs 1.

If the simulation arrives to the point where $\mathcal{A}$ outputs its guess b' (step 4 of the SD-GSD game), then $\mathcal{A}'$ outputs 0.

The advantage of $\mathcal{A}'$ is given by

$$\mathrm{Adv}_{\Pi}^{\mathrm{MI-EAV}}(\mathcal{A}') = \left|\Pr\left[\mathcal{A}' \to 1 \mid b = 0\right] - \Pr\left[\mathcal{A}' \to 1 \mid b = 1\right]\right|. \tag{3.2}$$

First we will show that

$$\Pr\left[\mathcal{A}' \to 1 \mid b = 0\right] \geq \frac{\Pr\left[Q_{\mathrm{s}} \wedge \overline{F_{\mathrm{DH}}}\right]}{N}. \tag{3.3}$$

Let $E = Q_{\mathrm{s}} \wedge \overline{F_{\mathrm{DH}}}$ and let E' the same event in the SD-GSD game simulated to $\mathcal{A}$ during an execution of $\mathrm{Game}_{\mathcal{A},\Pi_{\mathrm{s}}}^{\mathrm{MI-EAV}}$ with $b = 0$. In the following while showing (3.3) we will implicitly assume that $b = 0$ when referring to the game simulated to $\mathcal{A}$ by $\mathcal{A}'$. On a high level (3.3) holds because, as long as the game has not been aborted, the encyptions $\mathcal{A}$ receives from $\mathcal{A}'$ are indistinguishable from what it would get in the real SD-GSD game and we lose a factor $\frac{1}{N}$ due to guessing the node that triggered E. However, showing this requires a few steps.

Consider a modification of the SD-GSD game G_1 where the game is aborted whenever one of the following events occurs, where for all these events $\mathcal{A}'$ would also abort the simulation:

- $\mathcal{A}$ queries H_{DH} for a hidden Diffie-Hellman key
- $\mathcal{A}$ queries H_{gen} or H_{dep} for a hidden seed

(Since we are not interested in the output of the game we can define *aborting the game* as the game ending with output 0.) The game G_1 is something between the real SD-GSD game and what $\mathcal{A}'$ simulates to $\mathcal{A}$. The only difference in when G_1 aborts compared to the game simulated by $\mathcal{A}'$ is that we aren't paying attention to some specific node w remaining safe. Aborting the game in this way does not alter the probability of $\mathcal{A}$ triggering the event E in G_1, since in any case where the game is aborted either E or $\overline{E}$ will already be known to hold:

- If $\mathcal{A}$ queries H_{DH} for a hidden Diffie-Hellman key, then it triggers Q_{DH} and Q_{s} has not been triggered before since this would have caused the game to be aborted. Thus $\mathcal{A}$ triggered F_{DH} and $Q_{\mathrm{s}} \wedge \overline{F_{\mathrm{DH}}}$ cannot hold in this execution of the game.

- If $\mathcal{A}$ queries H_{gen} or H_{dep} for a hidden seed, then this triggers Q_{s}. Moreover, $\overline{F_{\text{DH}}}$ is also triggered at this moment since if F_{DH} had been triggered before (by triggering Q_{DH}), then the game would have already aborted. Thus $Q_{\text{s}} \wedge \overline{F_{\text{DH}}}$ was triggered.

Let E_1 the same event as E in the game G_1. As argued above we have

$$\Pr[E_1] = \Pr[E]. \tag{3.4}$$

Now consider a game G_2 which is a modification of game G_1 where at the beginning of the game $w_2 \leftarrow [n]$ is sampled and the game also aborts if $\mathcal{A}$ queries $\mathrm{corrupt}(u)$ for some node u such that w_2 is no longer safe, just as in the game simulated by $\mathcal{A}'$. The game G_2 is again something between the game G_1 and what $\mathcal{A}'$ simulates to $\mathcal{A}$. We also modify G_1 such that it also samples $w_1 \leftarrow [n]$ at the beginning of the game. This does not change the fact that (3.4) holds as the sampling of w_1 has no effect on the execution of the game. We further introduce a new random variable V to analyze the games G_1, G_2 and the game simulated by $\mathcal{A}'$, where

$$V = \begin{cases} 0 & \overline{E} \\ v & E \text{ was triggered at node } v \end{cases}.$$

Let V_1, V_2 and V' be the corresponding random variables in game G_1, game G_2 and the game simulated by $\mathcal{A}'$. Consider the probability $\Pr[V_1 = w_1 \mid E_1]$. The node w_1 is sampled independently and does not affect the execution of the game we have. Therefore, in an execution where E_1 occurs and the GSD graph has size n, we have a probability of $\frac{1}{n} \geq \frac{1}{N}$ of correctly guessing $V_1 = w_1$. Thus

$$\Pr[V_1 = w_1 \mid E_1] \geq \frac{1}{N}.$$

and also using (3.4) we get

$$\begin{aligned} \Pr[V_1 = w_1 \wedge E_1] &= \Pr[V_1 = w_1 \mid E_1] \cdot \Pr[E_1] \\ &\geq \frac{1}{N} \cdot \Pr[E]. \end{aligned} \tag{3.5}$$

Analogously to the argument used to show (3.4), we will show that

$$\Pr[V_1 = w_1 \wedge E_1] = \Pr[V_2 = w_2 \wedge E_2]. \tag{3.6}$$

The only difference from G_1 to G_2 is that G_2 aborts when w_2 is no longer safe. But if w_2 is no longer safe then we know that $V_2 \neq w_2$ (the game would have already aborted if w_2's seed had been queried while it was safe) and it is already decided that $V_2 = w_2 \wedge E_2$ does not hold. Thus the claim indeed holds.

We now show an analogous similar result comparing the game G_2 to the game simulated by $\mathcal{A}'$:

$$\Pr[V_2 = w_2 \wedge E_2] = \Pr\left[V' = w \wedge E'\right]. \tag{3.7}$$

Consider how G_2 differs from the game simulated by $\mathcal{A}'$. Both games abort at exactly the same events. They only differ in how $\mathcal{A}'$ answers queries encrypt(u, w) for any u. In G_2 such a query is answered with a ciphertext $\langle g^x, c\rangle$ where $x \leftarrow [|\mathbb{G}|], c \leftarrow \Pi_s.\text{Enc}_k(s_w)$ and $k = H_{\text{DH}}(pk_u^x)$. $\mathcal{A}'$ answers such a query with $\langle g^{x'}, c'\rangle$ where $x' \leftarrow [|\mathbb{G}|], c' \leftarrow \Pi_s.\text{Enc}_{k'}(s_w)$ and $k' \leftarrow \{0,1\}^\lambda$. Now notice that as long as the simulation of the game is ongoing, $pk_u^{x'}$ is a hidden Diffie-Hellman key and $\mathcal{A}$ has not queried $pk_u^{x'}$ to H_{DH}, because if it had then the game would already have aborted. Therefore, from $\mathcal{A}$'s view k' follows the same distribution as k. Thus, overall the game G_2 and the game simulated by $\mathcal{A}'$ are indistinguishable to $\mathcal{A}'$ and (3.7) holds.

Finally, notice that $\mathcal{A}'$ outputs 1 iff. the event $V' = w \wedge E'$ occurred. Then we have

$$\begin{aligned}
\Pr\left[\mathcal{A}' \to 1 \mid b = 0\right] &= \Pr\left[V' = w \wedge E'\right] \\
&\overset{(3.7)}{=} \Pr[V_2 = w_2 \wedge E_2] \\
&\overset{(3.6)}{=} \Pr[V_1 = w_1 \wedge E_1] \\
&\overset{(3.5)}{\geq} \frac{\Pr[E]}{N} \\
&= \frac{\Pr\left[Q_s \wedge \overline{F_{\text{DH}}}\,\right]}{N},
\end{aligned}$$

as promised.

Second we can more easily show that $\Pr[\mathcal{A}' \to 1 \mid b = 1]$ is negligible. In the SD-GSD game simulated to $\mathcal{A}$ during an execution of $\text{Game}_{\mathcal{A},\Pi_s}^{\text{MI-EAV}}$ with $b = 1$ the seed s_w is a random variable independent of any information given to $\mathcal{A}$:

- the game aborts when w becomes unsafe, so s_w cannot be learned by querying corrupt(w) or by querying $H_{\text{dep}}(p)$ for an unsafe node p where (p, w) is a seed dependency
- querying $H_{\text{dep}}(p)$ for a safe node p where (p, w) is a seed dependency results in the game being aborted
- with $b = 1$ queries encrypt(u, w) yield encryptions of s instead of s_w

Therefore, for any seed s' that $\mathcal{A}$ queries to H_{gen} or H_{dep} we have

$$\Pr\left[s_w = s'\right] = \frac{1}{2^\lambda}.$$

Thus, using the union bound we have

$$\Pr\left[\mathcal{A}' \to 1 \mid b = 1\right] \leq \frac{m_s}{2^\lambda}. \tag{3.8}$$

Combining (3.2), (3.3) and (3.8) we get

$$\begin{aligned} \mathrm{Adv}_{\Pi}^{\mathrm{MI-EAV}}(\mathcal{A}') &\geq \Pr\left[\mathcal{A}' \to 1 \mid b = 0\right] - \Pr\left[\mathcal{A}' \to 1 \mid b = 1\right] \\ &\geq \frac{\Pr\left[Q_s \wedge \overline{F_{\mathrm{DH}}}\right]}{N} - \frac{m_s}{2^\lambda}. \end{aligned} \tag{3.9}$$

Furthemore, going through the details yields that $\mathcal{A}'$ runs in time (the simulation of the SD-GSD game dominating the additional runtime)

$$\begin{aligned} t_{\mathcal{A}'} := \tilde{t} + \mathcal{O}\big(&m_s + m_{\mathrm{DH}} \\ &+ N \cdot (t_{H_{\mathrm{dep}}} + t_{\mathrm{sample}} + t_{H_{\mathrm{gen}}} + t_{\Pi_{\mathrm{DH}}.\mathrm{Gen}}) \\ &+ N^2 \cdot t_{\Pi_{\mathrm{DH}}.\mathrm{Enc}}\big). \end{aligned}$$

Using that $\delta \leq N, t_{\Pi_s.\mathrm{Gen}} = t_{\mathrm{sample}}, t_{\Pi_s.\mathrm{Enc}} \leq t_{\Pi_{\mathrm{DH}}.\mathrm{Enc}}$ (as encrypting with Π_{DH} involves an encryption with Π_s) and the definition of $\tilde{t}$, with appropriately chosen constants we have

$$t_{\mathcal{A}'} \leq t - \mathcal{O}(\delta \cdot (t_{\Pi_s.\mathrm{Gen}} + t_{\Pi_s.\mathrm{Enc}})).$$

By Lemma 3.7 Π_s is $(t - \mathcal{O}(\delta \cdot (t_{\Pi_s.\mathrm{Gen}+\Pi_s.\mathrm{Enc}})), \delta \cdot \varepsilon, \delta)$-MI-EAV secure, so

$$\mathrm{Adv}_{\Pi}^{\mathrm{MI-EAV}}(\mathcal{A}') \leq \delta \cdot \varepsilon. \tag{3.10}$$

Finally, if we now combine (3.9) and (3.10) we get

$$\begin{gathered} \frac{\Pr\left[Q_s \wedge \overline{F_{\mathrm{DH}}}\right]}{N} - \frac{m_s}{2^\lambda} \leq \delta \cdot \varepsilon \\ \Longleftrightarrow \\ \Pr\left[Q_s \wedge \overline{F_{\mathrm{DH}}}\right] \leq \delta \cdot N \cdot \varepsilon + \frac{m_s \cdot N}{2^\lambda}, \end{gathered}$$

as was to prove. □

Tighter EAV security for certain schemes

Definition 3.9 (Key-Rerandomizability) *Let Π a private-key encryption scheme. Π is* key-rerandomizable *if there exists a probabilistic algorithm* ReRan *running in time polynomial in ?, such that given $c \leftarrow \mathrm{Enc}_k(m)$ for any m and $k \leftarrow Gen()$,* text

Lemma 3.10

Chapter 4

Application to TreeKEM

4.1 The TreeKEM Protocol

Q: How much detail to provide on TreeKEM details?

4.2 Proving security for TreeKEM from SD-GSD security

TODO: Find the right CGKA definition.

TODO: Reduce TreeKEM security to GSD security.

Theorem 4.1

Appendix A

Dummy Appendix

You can defer lengthy calculations that would otherwise only interrupt the flow of your thesis to an appendix.

Bibliography

[1] Joël Alwen, Margarita Capretto, Miguel Cueto, Chethan Kamath, Karen Klein, Ilia Markov, Guillermo Pascual-Perez, Krzysztof Pietrzak, Michael Walter, and Michelle Yeo. Keep the dirt: Tainted treekem, adaptively and actively secure continuous group key agreement. Cryptology ePrint Archive, Paper 2019/1489, 2019. https://eprint.iacr.org/2019/1489.

[2] Richard Barnes, Benjamin Beurdouche, Raphael Robert, Jon Millican, Emad Omara, and Katriel Cohn-Gordon. The Messaging Layer Security (MLS) Protocol. RFC 9420, July 2023.

[3] Karthikeyan Bhargavan, Richard Barnes, and Eric Rescorla. TreeKEM: Asynchronous Decentralized Key Management for Large Dynamic Groups A protocol proposal for Messaging Layer Security (MLS). Research report, Inria Paris, May 2018.

[4] Giles Hogben. An important step towards secure and interoperable messaging. https://security.googleblog.com/2023/07/an-important-step-towards-secure-and.html, 2023. Accessed: 2023-11-01.

[5] IETF. Support for mls. https://www.ietf.org/blog/support-for-mls-2023/, 2023. Accessed: 2023-11-01.

[6] Saurabh Panjwani. Tackling adaptive corruptions in multicast encryption protocols. In *Proceedings of the 4th Conference on Theory of Cryptography*, TCC'07, page 21–40, Berlin, Heidelberg, 2007. Springer-Verlag.

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

Authored by (in block letters):
For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**

With my signature I confirm that
- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	**Signature(s)**

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.