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 [7], 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 ([7]) 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 $\mathcal{O}((n \cdot Q)^2)$ (**TODO:** Is $n \cdot Q$ really correct?), 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

2.1 Notation

We will use the following notation throughout:

- We write $x \leftarrow S$ to say that x is sampled uniformly at random from the finite set S
- If $\mathbb{G}$ is a cyclic group and g a generator, then
  - We write the group operation in $\mathbb{G}$ multiplicatively
  - h^{-1} denotes the inverse of $h \in \mathbb{G}$
  - $\log_g(h)$ denotes the unique $x \in [|\mathbb{G}|]$ such that $g^x = h$
- We write $\mathcal{A} \rightarrow b$ to denote the event that a probabilistic polynomial time adversary $\mathcal{A}$ outputs the bit b when playing a game where it must output a bit in the end.

2.2 Basic definitions

The definitions presented in this section were taken from [6].

2.2.1 Encryption schemes

Private-key encryption schemes

Q: Ok to copy more or less word for word?

Definition 2.1 *A* private-key encryption scheme Π *consists of three probabilistic polynomial-time algorithms* $(\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec})$ *such that:*

1. *The* key-generation algorithm Gen *takes as input* 1^n *(in unary) where* n *is the security parameter and outputs a key* k*. We will assume the security parameter to be fixed and write* $k \leftarrow \mathrm{Gen}()$.

2. *The* encryption algorithm Enc *takes as input a key k and a plaintext message $m \in \{0,1\}^*$, or $m \in \{0,1\}^{\leq \eta}$ for some η if the message space is finite, and outputs a ciphertext c. We write this as $c \leftarrow \mathrm{Enc}_k(m)$.*

3. *The* decryption algorithm Dec *takes as input a key k and a ciphertext c, and outputs a message m or $\perp$ (denoting an error). We write this as $m = \mathrm{Dec}_s k(c)$.*

We may also refer to algorithm X by $\Pi.X$ for $X \in \{\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec}\}$.

It is required that for every n, every key k output by Gen*, and every message m, it holds that $\mathrm{Dec}_k(\mathrm{Enc}_k(m)) = m$.*

Public-key encryption schemes

Definition 2.2 *A* public-key encryption scheme Π *consists of three probabilistic polynomial-time algorithms* $(\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec})$ *such that:*

1. *The* key-generation algorithm Gen *takes as input 1^n (in unary) where n is the security parameter and outputs a pair of keys (pk, sk) (a public and private key). We will assume the security parameter to be fixed and write $(pk, sk) \leftarrow \mathrm{Gen}()$.*

2. *The* encryption algorithm Enc *takes as input a public key pk and a plaintext message $m \in \mathcal{M}$ where $\mathcal{M}$ is the message space and outputs a ciphertext c. We write this as $c \leftarrow \mathrm{Enc}_{pk}(m)$.*

3. *The* decryption algorithm Dec *takes as input a private key sk and a ciphertext c, and outputs a message m or $\perp$ (denoting an error). We write this as $m = \mathrm{Dec}_{sk}(c)$.*

We may also refer to algorithm X by $\Pi.X$ for $X \in \{\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec}\}$.

It is required that for every n, every key (pk, sk) output by Gen*, and every message m, it holds that $\mathrm{Dec}_{sk}(\mathrm{Enc}_{pk}(m)) = m$.*

2.2.2 Security definitions

Definition 2.3 (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 2.4 (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*

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

We will make use a slightly weaker form of security called indistinguishability in the presence of an eavesdropper and will refer to it as EAV security. It is identical to IND-CPA security with the sole exception that the adversary does not have access to an encryption oracle.

Definition 2.5 (The EAV game) *Let* Π *a private-key encryption scheme. Define the game* $\mathrm{Game}_{\mathcal{A},\Pi}^{\mathrm{EAV}}$ *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 2.6 (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 2.7 *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. □

Definition 2.8 (The Decisional Diffie-Hellman (DDH) problem) *Let* $\mathbb{G}$ *a cyclic group and* g *a generator. Define the game* $\mathrm{Game}_{\mathcal{A},(\mathbb{G},g)}^{\mathrm{DDH}}$ *for an adversary* $\mathcal{A}$*:*

1. *Exponents* $x, y \leftarrow [|\mathbb{G}|]$ *and a bit* $b \leftarrow \{0, 1\}$ *are sampled.*
2. *The adversary* $\mathcal{A}$ *is given* $h_1 := g^x, h_2 := g^y$ *and*

   $$k = \begin{cases} g^{x \cdot y} & b = 0 \\ \tilde{k} & b = 1 \end{cases}$$

   where $\tilde{k} \leftarrow \mathbb{G}$*.*
3. $\mathcal{A}$ *outputs a bit* b'*. The output of the game is defined to be* 1 *if* $b' = b$*, and* 0 *otherwise.*

Definition 2.9 (Hardness of the DDH problem) *The DDH problem is* (t, ε)-hard in $\mathbb{G}$ *with the generator* g *if for any adversary* $\mathcal{A}$ *running in time* t *we have*

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

2.2.3 The Random Oracle Model

We will work in the commonly used Random Oracle Model (ROM) to prove our results. The ROM introduces the concept of a *random oracle*. A random oracle is a function $H : A \to B$ where certain assumptions are made about what an adversary $\mathcal{A}$ knows about H and how it interacts with it:

- From $\mathcal{A}$'s perspective, H is a black-box function. The only way for $\mathcal{A}$ to interact with H is for it to provide a value $a \in A$ and get back $H(a)$, and this is the only way for $\mathcal{A}$ to learn $H(a)$. We say that $\mathcal{A}$ *queries* $H(a)$ or that $\mathcal{A}$ *queries* H *for* a.

- From $\mathcal{A}$'s perspective, H is a random variable, uniformly sampled from the set of all functions from A to B. Thus, if $\mathcal{A}$ queries H for some $a \in A$ that it has not queried before, the value $H(a)$ is a random variable uniformly distributed in B from $\mathcal{A}$'s perspective.

We do not rely on the property known as "programmability" in this work.

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 the results to be applied to TreeKEM more directly. We motivate the modifications made later in Section 4 on page 27.

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* $(sk_v, pk_v) = \mathrm{Gen}(H_{\mathrm{gen}}(s_v))$. **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:*
   - $\mathrm{reveal}(v)$ *for* $v \in [n]$: $\mathcal{A}$ *is given* pk_v.
   - $\mathrm{encrypt}(u, v)$ *for* $u, v \in [n], u \neq v, (u, v) \notin E$: (u, v) *is added to* E *and* $\mathcal{A}$ *is given* $c \leftarrow \mathrm{Enc}_{pk_u}(s_v)$.
   - $\mathrm{corrupt}(v)$ *for* $v \in [n], v \notin \mathcal{C}$: $\mathcal{A}$ *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 call their seeds* 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*

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

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:** *This avoids multiple sources for single target.*

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, our definition of security is centered around the choice of the encryption scheme Π.

Definition 3.2 (SD-GSD security) *A public-key encryption scheme* Π *is* $(t, \varepsilon, N, \delta)$-SD-GSD secure *if there exist instantiations of the KDFs* H_{gen} *and* H_{dep} *such that (***TODO:** *This formulation with KDFs makes sense?) 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*

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

3.2 Proving SD-GSD security for DHIES

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 $\Pi_s.\mathrm{Gen}$ samples a key uniformly at random from $\{0,1\}^\kappa$. 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_{gen}, H_{dep} and H_{DH} are modelled as random oracles, then Π_{DH} is $(\tilde{t}, \tilde{\varepsilon}, N, \delta)$-SD-GSD secure with*

$$\tilde{\varepsilon} = 2 \cdot (\delta + 1) \cdot N \cdot \varepsilon + \frac{2 \cdot m_{\mathrm{DH}} \cdot N^2}{|\mathbb{G}|} + \frac{m_{\mathrm{s}} \cdot N}{2^{\lambda-1}},$$

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}, and with

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

where the various variables denote the following

- $t_{H_{\mathrm{gen}}}$: *time to evaluate H_{gen} on $s \in \{0,1\}^\lambda$*
- $t_{H_{\mathrm{dep}}}$: *time to evaluate H_{dep} on $s \in \{0,1\}^\lambda$*
- t_{sample}: *time to sample a bit*
- $t_{\Pi_{\mathrm{DH}}.\mathrm{Enc}}$: *time to encrypt $s \in \{0,1\}^\lambda$ with Π_{DH}*
- $t_{\Pi_{\mathrm{DH}}.\mathrm{Gen}}$: *runtime of $\Pi_{\mathrm{DH}}.\mathrm{Gen}$*
- t_{op}: *time to perform the group operation in $\mathbb{G}$*
- γ: *length of encoding of a group element $\mathbb{G}$ (used in Lemma 3.7)*

Intuition Consider an arbitrary SD-GSD adversary $\mathcal{A}$. For an execution of $\mathrm{Game}^{\mathrm{SD-GSD}}_{\mathcal{A},\Pi_{\mathrm{DH}}}$ we say "*$\mathcal{A}$ wins*" to denote the event $\mathrm{Game}^{\mathrm{SD-GSD}}_{\mathcal{A},\Pi_{\mathrm{DH}}} = 1$. 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 $x \in \{\mathrm{gen}, \mathrm{dep}\}$. Then we can write

$$\begin{aligned}\Pr[\mathcal{A} \text{ wins}] &= \Pr[\mathcal{A} \text{ wins} \wedge Q_{\mathrm{dep}}] + \Pr[\mathcal{A} \text{ wins} \wedge \overline{Q_{\mathrm{dep}}}] \\ &\leq \Pr[\mathcal{A} \text{ wins} \wedge Q_{\mathrm{dep}}] + \Pr[\mathcal{A} \text{ wins} \mid \overline{Q_{\mathrm{dep}}}] \\ &\overset{(\dagger)}{=} \Pr[\mathcal{A} \text{ wins} \wedge Q_{\mathrm{dep}}] + \frac{1}{2} \\ &\leq \Pr[Q_{\mathrm{dep}}] + \frac{1}{2} \\ &\leq \Pr[Q_{\mathrm{s}}] + \frac{1}{2},\end{aligned} \tag{3.1}$$

where $Q_{\mathrm{s}} := Q_{\mathrm{gen}} \cup Q_{\mathrm{dep}}$ (s for *seed*). Step (†) intuitively holds because without having queried H_{dep} for any hidden seed, in particular s_v, $H_{\mathrm{dep}}(s_v)$ is a uniformly random value from $\mathcal{A}$'s perspective. Therefore, it can do no better than guessing to distinguish $H_{\mathrm{dep}}(s_v)$ from s.

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 $\Pr[Q_{\mathrm{s}}]$. When the adversary first triggers Q_{s} by querying the seed of some safe node w, (with overwhelming probability w will be the only node with this seed and) it can only have learned the seed through encryptions $c_1 \leftarrow \Pi_{\mathrm{DH}}.\mathrm{Enc}_{pk_{u_1}}(s_w), \ldots, c_d \leftarrow \Pi_{\mathrm{DH}}.\mathrm{Enc}_{pk_{u_d}}(s_w)$ where $(u_1, w), \ldots, (u_d, w)$ are edges in the GSD graph (obtained through corresponding queries $\mathrm{encrypt}(u_1, w), \ldots, \mathrm{encrypt}(u_d, w)$). The only other potential source of information about s_w would be a seed dependency (p, w), but this tells $\mathcal{A}$ nothing: Since w is safe, p would also be safe and $H_{\mathrm{dep}}(s_p)$ cannot have been queried due to the assumption that w was the first node to trigger Q_{s}. Without having queried $H_{\mathrm{dep}}(s_p)$, by virtue of H_{dep} being a random oracle s_w has the same distribution as a seed without a dependency from $\mathcal{A}$'s perspective (uniformly random).

TODO: Add plot illustrating edges in GSD graph and a potential seed dependency.

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 $x_i = \log_g(pk_{u_i})$ (where g is the generator of $\mathbb{G}$ being used in Π_{DH}). Each encryption c_i is a tuple of the form $\langle g^{y_i}, \Pi_s.\mathrm{Enc}_{k_i}(s_w)\rangle$ where $y_i \leftarrow [|\mathbb{G}|], k_i = H_{\mathrm{DH}}(g^{x_i \cdot y_i})$. 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 as before if w is safe, then so are the nodes u_i. Thus the adversary has not learned the exponent x_i through querying $H_{\mathrm{gen}}(s_{u_i})$ 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 $\Pi_s.\mathrm{Enc}_{k_1}(s_w), \ldots, \Pi_s.\mathrm{Enc}_{k_d}(s_w)$.

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 $k = pk_u^{y_{u,v}}$, where (u,v) is an edge in the GSD graph, u is safe and $y_{u,v}$ is the exponent chosen in the DHIES encryption of s_v (i.e. $\mathcal{A}$ was given a ciphertext of the form $\langle g^{y_{u,v}}, \ldots \rangle$ when it queried encrypt(u,v)). 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 the event Q_{s} into two cases as motivated above:

$$\Pr[Q_{\mathrm{s}}] = \Pr[Q_{\mathrm{s}} \wedge F_{\mathrm{DH}}] + \Pr\left[Q_{\mathrm{s}} \wedge \overline{F_{\mathrm{DH}}}\,\right].$$

We bound $\Pr[Q_{\mathrm{s}} \wedge F_{\mathrm{DH}}]$ and $\Pr\left[Q_{\mathrm{s}} \wedge \overline{F_{\mathrm{DH}}}\,\right]$ in Lemma 3.11 and Lemma 3.7, respectively. Overall this gives us a bound on the advantage of $\mathcal{A}$ using (3.1). (To be precise, the event $Q_{\mathrm{s}} \wedge F_{\mathrm{DH}}$ is a superset of the first scenario described further above. However, the argument applied in Lemma 3.11 gives the same bound for either event and this more general event has the advantage of being simpler.)

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 (3.1). Note that by the rules imposed on the adversary in the SD-GSD game, the challenge node v is safe and its seed s_v thus indeed hidden. If Q_{dep} does not hold, then $\mathcal{A}$ has not queried H_{dep} for s_v and, by virtue of H_{dep} being a random oracle, $H_{\mathrm{dep}}(s_v)$ is a uniformly distributed value in $\{0,1\}^\lambda$ from $\mathcal{A}$'s perspective. The value s follows the same distribution. Thus, $\mathcal{A}$ behaves the same when given either $r = s$ or $r = H_{\mathrm{dep}}(s_v)$ and

$$\begin{aligned}\Pr\left[\mathcal{A} \to 1 \mid \overline{Q_{\mathrm{dep}}}, b = 1\right] &= \Pr\left[\mathcal{A} \to 1 \mid \overline{Q_{\mathrm{dep}}}, r = s\right] \\ &= \Pr\left[\mathcal{A} \to 1 \mid \overline{Q_{\mathrm{dep}}}, r = H_{\mathrm{dep}}(s_v)\right] \\ &= \Pr\left[\mathcal{A} \to 1 \mid \overline{Q_{\mathrm{dep}}}, b = 0\right].\end{aligned} \tag{3.2}$$

Therefore

$$\begin{aligned}\Pr\left[\mathcal{A} \text{ wins} \mid \overline{Q_{\mathrm{dep}}}\,\right] = {} & \Pr\left[\mathcal{A} \to 1 \mid \overline{Q_{\mathrm{dep}}}, b = 1\right] \cdot \frac{1}{2} \\ & + \Pr\left[\mathcal{A} \to 0 \mid \overline{Q_{\mathrm{dep}}}, b = 0\right] \cdot \frac{1}{2} \\ \overset{(3.2)}{=} {} & \Pr\left[\mathcal{A} \to 1 \mid \overline{Q_{\mathrm{dep}}}, b = 0\right] \cdot \frac{1}{2} \\ & + \Pr\left[\mathcal{A} \to 0 \mid \overline{Q_{\mathrm{dep}}}, b = 0\right] \cdot \frac{1}{2} \\ = {} & \frac{1}{2}.\end{aligned}$$

By Lemma 3.11 on page 22 we have

$$\Pr[Q_{\mathrm{s}} \wedge F_{\mathrm{DH}}] \leq N \cdot \varepsilon + \frac{m_{\mathrm{DH}} \cdot N^2}{|\mathbb{G}|}.$$

and by Lemma 3.7 on page 14 we have

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

so we know that

$$\Pr[Q_{\mathrm{s}}] \leq (\delta + 1) \cdot N \cdot \varepsilon + \frac{m_{\mathrm{DH}} \cdot N^2}{|\mathbb{G}|} + \frac{m_{\mathrm{s}} \cdot N}{2^{\lambda}}.$$

Then by (3.1)

$$\Pr[\mathcal{A} \text{ wins}] \leq (\delta + 1) \cdot N \cdot \varepsilon + \frac{m_{\mathrm{DH}} \cdot N^2}{|\mathbb{G}|} + \frac{m_{\mathrm{s}} \cdot N}{2^{\lambda}} + \frac{1}{2},$$

so

$$\mathrm{Adv}_{\Pi}^{\mathrm{SD-GSD}}(\mathcal{A}) \leq 2 \cdot \left((\delta + 1) \cdot N \cdot \varepsilon + \frac{m_{\mathrm{DH}} \cdot N^2}{|\mathbb{G}|} + \frac{m_{\mathrm{s}} \cdot N}{2^{\lambda}} \right) = \tilde{\varepsilon}. \qquad \square$$

3.2.1 Reducing to EAV security

TODO: Motivate MI-EAV security by relating to intuition of Lemma 3.3.

Definition 3.4 (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 3.5 (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 is clear that one should be able to use an analogous 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 $H_0, \dots, H_q$ where in the game H_i the first i encryption queries encrypt the second message and the remaining $q - i$ queries encrypt the first message. 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. The adversary $\mathcal{A}'$ simulates H_{i-1} or H_i to $\mathcal{A}$ depending on whether the ciphertext received from the (single-query) IND-CPA challenger, which gets passed on as the response to the i-th query, encrypts the first or the second message from the i-th pair of messages. $\mathcal{A}'$ then uses the encryption oracle 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 $\mathcal{A}'$ sampling the new keys itself.

Lemma 3.6 *Let Π a private-key encryption scheme with finite message space. Let $t_{\text{Gen}}, t_{\text{Enc}}$ upper bounds for the runtime of $\Pi.\text{Gen}$ and $\Pi.\text{Enc}$, 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 $\tilde{t} = t - \mathcal{O}(q \cdot (t_{\text{Gen}} + t_{\text{Enc}}))$.*

Q: Move proof to appendix?

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 $H_0, \dots, H_q$ 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 $c_1, \dots, c_q$ 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 $k_j \leftarrow \Pi.\text{Gen}()\ \forall j$.

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

$$\begin{aligned} |\Pr[\mathcal{A} \to 1 \mid H_{i-1}] - & \Pr[\mathcal{A} \to 1 \mid H_i] \\ &= \left|\Pr\left[\mathcal{A}' \to 1 \mid b = 0\right] - \Pr\left[\mathcal{A}' \to 1 \mid b = 1\right]\right| \\ &= \mathrm{Adv}_{\Pi}^{\mathrm{EAV}}(\mathcal{A}') \\ &\leq \varepsilon \end{aligned} \tag{3.3}$$

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

$$\begin{aligned} \mathrm{Adv}_{\Pi}^{\mathrm{MI-EAV}}(\mathcal{A}) &= |\Pr[\mathcal{A} \to 1 \mid b = 0] - \Pr[\mathcal{A} \to 1 \mid b = 1]| \\ &= \left|\Pr[\mathcal{A} \to 1 \mid H_0] - \Pr\left[\mathcal{A} \to 1 \mid H_q\right]\right| \\ &= \left|\sum_{i=1}^{q} \Pr[\mathcal{A} \to 1 \mid H_{i-1}] - \Pr[\mathcal{A} \to 1 \mid H_i]\right| \\ &\leq \sum_{i=1}^{q} |\Pr[\mathcal{A} \to 1 \mid H_{i-1}] - \Pr[\mathcal{A} \to 1 \mid H_i]| \\ &\overset{(3.3)}{\leq} q \cdot \varepsilon. \end{aligned}$$

□

Lemma 3.7 *Recall the assumptions, variables and events from 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*

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

Intuition By Lemma 3.6 on page 13 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 $Q_s \wedge \overline{F_{DH}}$ by querying the seed of some safe node w. As intended, 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) for any u contain encryptions of s_w under independent, uniformly random keys using Π_s. Moreover, as already argued once, $\mathcal{A}$ has learned nothing from a potential seed dependency (p, w), so these encryptions are everything $\mathcal{A}$ 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 a node z whose seed $\mathcal{A}$ may query. Next we give the MI-EAV challenger s_z 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, z) for any u. Now consider the behavior of $\mathcal{A}$ depending on which seed the challenger chooses to encrypt:

- If the challenger chooses to encrypt s_z, then $\mathcal{A}$ will trigger the event $Q_s \wedge \overline{F_{DH}}$ with the same probability as before and if we guessed z correctly (i.e. $z = w$) we can detect whether $Q_s \wedge \overline{F_{DH}}$ gets triggered (by checking if $H_{\text{gen}}(s_z)$ or $H_{\text{dep}}(s_z)$ was queried by $\mathcal{A}$ during the simulation).
- If the challenger chooses to encrypt s, then $\mathcal{A}$ receives no information about s_z and has negligible probability of querying it.

Thus the advantage of the adversary is about $\Pr\left[Q_s \wedge \overline{F_{DH}}\right]/N$, where the factor $1/N$ arises from guessing z, and using that Π_s is MI-EAV secure we can bound this probability. Since we are only interested in checking whether the event was triggered for z, the adversary can abort when this is no longer possible (z is corrupted, some other hidden seed is queried, etc.).

Proof (of Lemma 3.7) 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 the set of edges and 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 $c_1, \ldots, c_\delta$ 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) for any u, $\mathcal{A}'$ replies with $\langle g^x, c_i \rangle$ where $x \leftarrow [|\mathbb{G}|]$ and i is the 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 $H_{\mathrm{gen}}(s_w)$ or $H_{\mathrm{dep}}(s_w)$, $\mathcal{A}'$ 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 (step 4 of the SD-GSD game), then $\mathcal{A}'$ outputs 0.

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

$$\mathrm{Adv}_{\Pi}^{\mathrm{MI-EAV}}(\mathcal{A}') = \left|\Pr\left[\mathcal{A}' \to 1 \mid b = 0\right] - \Pr\left[\mathcal{A}' \to 1 \mid b = 1\right]\right|, \tag{3.4}$$

where b is the bit sampled by the MI-EAV challenger.

First, we will show that

$$\Pr\left[\mathcal{A}' \to 1 \mid b = 0\right] \geq \frac{\Pr\left[Q_{\mathrm{s}} \wedge \overline{F_{\mathrm{DH}}}\right]}{N}. \tag{3.5}$$

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 $\mathrm{Game}_{\mathcal{A}',\Pi_s}^{\mathrm{MI-EAV}}$ with $b = 0$. In the following while showing (3.5) we will implicitly assume that $b = 0$ when referring to the game simulated to $\mathcal{A}$ by $\mathcal{A}'$. On a high level (3.5) holds due to the fact that 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 get a factor $\frac{1}{N}$ from 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 either case when the game is aborted either E or $\overline{E}$ is already 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 $Q_{\text{s}} \wedge \overline{F_{\text{DH}}}$ 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, $\overline{F_{\text{DH}}}$ also holds at this moment since the game would have aborted earlier if Q_{DH} had already been triggered. Thus $Q_{\text{s}} \wedge \overline{F_{\text{DH}}}$ holds.

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

$$\Pr[E_1] = \Pr[E]. \tag{3.6}$$

Now consider a game G_2 which is a modification of the 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 $\text{corrupt}(u)$ 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 (3.6) holds as the sampling of w_1 has no effect on the execution of the game.

Let E_2 and E' the events corresponding to E in the game G_2 and the game simulated by $\mathcal{A}'$, respectively. We further introduce a new random variable W to analyze each game where

$$W = \begin{cases} 0 & \overline{E} \\ x & E \text{ was triggered at node } x \end{cases}$$

(if x is not unique we choose the node with smallest identifier). Let W_1, W_2 and W' be the corresponding random variables in game G_1, game G_2 and the game simulated by $\mathcal{A}'$. Consider the probability $\Pr[W_1 = w_1 \mid E_1]$. The node w_1 is sampled independently and does not affect the execution of the game. Therefore, in an execution where E_1 occurs and the GSD graph has size n (so $W_1 \in [n]$), we correctly guess $W_1 = w_1$ with probability exactly $\frac{1}{n} \geq \frac{1}{N}$. Thus

$$\Pr[W_1 = w_1 \mid E_1] \geq \frac{1}{N}$$

and combining this with (3.6) we get

$$\begin{aligned} \Pr[W_1 = w_1] &= \Pr[W_1 = w_1 \wedge E_1] \\ &= \Pr[W_1 = w_1 \mid E_1] \cdot \Pr[E_1] \\ &\geq \frac{1}{N} \cdot \Pr[E]. \end{aligned} \tag{3.7}$$

Analogously to the argument used to justify (3.6), we can argue that

$$\Pr[W_1 = w_1] = \Pr[W_2 = w_2]. \tag{3.8}$$

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 $W_2 \neq w_2$ (if $W_2 = w_2$ the game would have already aborted when w_2's seed was queried while it was safe). Thus (3.6) indeed holds.

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

$$\Pr[W_2 = w_2] = \Pr\left[W' = w\right]. \tag{3.9}$$

Consider how G_2 differs from the game simulated by $\mathcal{A}'$. Both games abort at exactly the same events (verify this! **Q:** Ok to add such a note for the reader?). They only differ in how $\mathcal{A}'$ answers queries $\mathsf{encrypt}(u, w)$ for any u. In G_2 such a query is answered with a ciphertext $\langle g^x, c\rangle$ where $x \leftarrow [|\mathbb{G}|], c \leftarrow \Pi_s.\mathsf{Enc}_k(s_w)$ and $k = H_{\mathrm{DH}}(pk_u^x)$. $\mathcal{A}'$ answers such a query with $\langle g^{x'}, c'\rangle$ where $x' \leftarrow [|\mathbb{G}|], c' \leftarrow \Pi_s.\mathsf{Enc}_{k'}(s_w)$ and $k' \leftarrow \{0,1\}^\kappa$. Now notice that as long as the game G_2 is ongoing, pk_u^x is a hidden Diffie-Hellman key and $\mathcal{A}$ has not queried pk_u^x to H_{DH}. If it had, then the game would have already 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 (3.9) holds.

Finally, notice that if the event $W' = w$ occurred, then $\mathcal{A}'$ outputs 1. Then we have

$$\begin{aligned}
\Pr\left[\mathcal{A}' \to 1 \mid b = 0\right] &\geq \Pr\left[W' = w\right] \\
&\overset{(3.9)}{=} \Pr[W_2 = w_2] \\
&\overset{(3.8)}{=} \Pr[W_1 = w_1] \\
&\overset{(3.7)}{\geq} \frac{\Pr[E]}{N} \\
&= \frac{\Pr\left[Q_s \wedge \overline{F_{\mathrm{DH}}}\right]}{N},
\end{aligned}$$

as promised.

Second, returning to (3.4), 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 $\mathsf{Game}_{\mathcal{A}',\Pi_s}^{\mathrm{MI-EAV}}$ 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 $\mathsf{corrupt}(w)$ or by querying $H_{\mathrm{dep}}(s_p)$ for an unsafe node p where (p, w) is a seed dependency

- querying $H_{\text{dep}}(s_p)$ for a safe node p where (p, w) is a seed dependency results in the game being aborted and by virtue of H_{dep} being a random oracle, from $\mathcal{A}$'s perspective s_w follows the same distribution regardless of whether there is a seed dependency (p, w) or not
- 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

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

Thus, by a union bound we have

$$\Pr[\mathcal{A}' \to 1 \mid b = 1] \leq \frac{m_{\text{s}}}{2^\lambda}. \tag{3.10}$$

Combining (3.4), (3.5) and (3.10) we get

$$\begin{aligned} \text{Adv}_{\Pi}^{\text{MI-EAV}}(\mathcal{A}') &\geq \Pr[\mathcal{A}' \to 1 \mid b = 0] - \Pr[\mathcal{A}' \to 1 \mid b = 1] \\ &\geq \frac{\Pr[Q_{\text{s}} \wedge \overline{F_{\text{DH}}}]}{N} - \frac{m_{\text{s}}}{2^\lambda}. \end{aligned} \tag{3.11}$$

Furthemore, going through the details yields that $\mathcal{A}'$ runs in time

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

(the simulation of the SD-GSD game dominating the additional runtime). Using that $t_{\text{op}} = \Omega(\gamma), \delta \leq N, t_{\Pi_{\text{s}}.\text{Gen}} = \mathcal{O}(\kappa \cdot t_{\text{sample}}), t_{\Pi_{\text{s}}.\text{Enc}} \leq t_{\Pi_{\text{DH}}.\text{Enc}}$ (as encrypting with Π_{DH} involves an encryption with Π_s) and the definition of $\tilde{t}$, with appropriately chosen constants we have

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

By Lemma 3.6 Π_s is $(t - \mathcal{O}(\delta \cdot (t_{\Pi_{\text{s}}.\text{Gen}} + \Pi_{\text{s}}.\text{Enc})), \delta \cdot \varepsilon, \delta)$-MI-EAV secure, so

$$\text{Adv}_{\Pi}^{\text{MI-EAV}}(\mathcal{A}') \leq \delta \cdot \varepsilon. \tag{3.12}$$

Finally, if we now combine (3.11) and (3.12) we get

$$\begin{gathered} \frac{\Pr[Q_{\text{s}} \wedge \overline{F_{\text{DH}}}]}{N} - \frac{m_{\text{s}}}{2^\lambda} \leq \delta \cdot \varepsilon \\ \iff \\ \Pr[Q_{\text{s}} \wedge \overline{F_{\text{DH}}}] \leq \delta \cdot N \cdot \varepsilon + \frac{m_{\text{s}} \cdot N}{2^\lambda}, \end{gathered}$$

as was to prove. □

Tighter MI-EAV security for certain schemes

In our reduction from MI-EAV security to EAV security (Lemma 3.6) we applied a general hybrid argument. It is also tempting to try a more direct approach. The EAV and MI-EAV games seem less far apart than IND-CPA for single and multiple encryptions: All additional encrpytions in the MI-EAV game encrypt the same message, with the only difference being that each encryption is performed using a fresh key. If only we could take a single encryption $c \leftarrow \mathrm{Enc}_k(m)$ and from it produce several encryptions $c_i \leftarrow \mathrm{Enc}_{k_i}(m)$ for $k_i \leftarrow \mathrm{Gen}()$ (without knowing k or m), then the additional encryptions would leak no new information to the adversary, and we would have a tight bound on MI-EAV security from EAV security. There is a simple EAV secure scheme that achieves the above property: the one-time pad. Given an encryption $c = k \oplus m$, we can simply sample $k' \leftarrow \{0,1\}^\kappa$ and compute the ciphertext $c' = c \oplus k' = (k \oplus k') \oplus m$, an encryption of m under the uniformly random key $k \oplus k'$. In the following, we formalize this property of a private-key encryption scheme and use it to prove the desired bound on MI-EAV security.

Definition 3.8 (Key-rerandomizability) *Let Π a private-key encryption scheme with security parameter κ. Π is* key-rerandomizable *if there exists a probabilistic polynomial time algorithm* ReRan *achieving the following: Given $c \leftarrow \mathrm{Enc}_k(m)$ for any fixed message m in the message space and $k \leftarrow \mathrm{Gen}()$, the output $c' \leftarrow \mathrm{ReRan}(c)$ follows the same distribution as the process of sampling $k' \leftarrow \mathrm{Gen}()$ and computing a ciphertext $\mathrm{Enc}_{k'}(m)$. The runtime must be polynomial in κ and the length of the ciphertext c.*

Example As outlined above, the one-time pad is an example of a key-rerandomizable encryption scheme.

Q: Is there a key-rerandomizable IND-CPA secure scheme? If yes, this would imply a key-rerandomizable AE scheme using the encrypt-then-authenticate paradigm, since a rerandomized tag can easily produced for the ciphertext by sampling a fresh MAC key.

The key idea underlying the proof of the following Lemma was already provided at the beginning of this section.

Lemma 3.9 *Let Π a key-rerandomizable private-key encryption scheme with finite message space. Let* ReRan *the corresponding algorithm to rerandomize ciphertexts and t_{ReRan} an upper bound for the runtime of* ReRan. *If Π is (t, ε)-EAV secure, then for all $q \in \mathbb{N}$, Π is $(\tilde{t}, \varepsilon, q)$-MI-EAV secure with $\tilde{t} = t - \mathcal{O}(q \cdot t_{\mathrm{ReRan}})$.*

Proof Note that since the message space and thus the ciphertext space is finite, the runtime of ReRan is indeed bounded. Let $\mathcal{A}$ an MI-EAV adversary running in time $\tilde{t}$ and making at most q queries. We construct an EAV adversary $\mathcal{A}'$ that behaves as follows:

1. $\mathcal{A}'$ runs $\mathcal{A}$ to get the number of queries q and messages m_0, m_1.
2. $\mathcal{A}'$ gives m_0, m_1 to the challenger and receives. Let c_1 the ciphertext it gets back.
3. $\mathcal{A}'$ computes ciphertexts $c_2 \leftarrow \mathrm{ReRan}(c_1), \ldots, c_q \leftarrow \mathrm{ReRan}(c_1)$ (with independent runs of ReRan).
4. $\mathcal{A}'$ gives the ciphertexts $c_1, \ldots, c_q$ to $\mathcal{A}$.
5. $\mathcal{A}'$ outputs whatever bit $\mathcal{A}$ outputs.

We apply the properties of ReRan given in Definition 3.8 to show that the game simulated to $\mathcal{A}$ is distributed identically to the MI-EAV game. For this we need only show that the ciphertexts $c_1, \ldots, c_q$ given to $\mathcal{A}$ in the simulation are distributed identically to the ciphertexts $c'_1, \ldots, c'_q$ that $\mathcal{A}$ would get in the real MI-EAV game. It is immediate that c_1 is distributed identically to c'_1. Now let $i \in \{2, \ldots, q\}$. By Definition 3.8 $\mathrm{ReRan}(c)$ outputs a ciphertext encrypting m_b (where b is the bit chosen by the EAV challenger) distributed identically to a ciphertext encrypting m_b output by the MI-EAV challenger. Thus, indeed for any i, c_i is distributed identically to c'_i and the claim holds. Therefore

$$\mathrm{Adv}_{\Pi}^{\mathrm{MI-EAV}}(\mathcal{A}) = \mathrm{Adv}_{\Pi}^{\mathrm{EAV}}(\mathcal{A}'). \tag{3.13}$$

Because $\mathcal{A}'$ is an EAV adversary running in time $\tilde{t} + \mathcal{O}(q \cdot t_{\mathrm{ReRan}}) = t$ we know that

$$\mathrm{Adv}_{\Pi}^{\mathrm{EAV}}(\mathcal{A}') \leq \varepsilon,$$

which together with (3.13) concludes the proof. □

By assuming a key-rerandomizable encryption scheme and applying Lemma 3.9 on the facing page instead of the hybrid argument (Lemma 3.6) in the proof of Lemma 3.7, we can drop the δ factor in the bound. This also allows us to drop the δ factor in Theorem 3.3 on page 9.

Corollary 3.10 *Recall the setting of Theorem 3.3. If the private-key encryption scheme Π_s is additionally key-rerandomizable, then the bound in Lemma 3.7 can be improved to*

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

and the bound $\tilde{\varepsilon}$ on the success probability of an SD-GSD adversary thus improved to

$$\tilde{\varepsilon} = 4 \cdot N \cdot \varepsilon + \frac{2 \cdot m_{\mathrm{DH}} \cdot N^2}{|\mathbb{G}|} + \frac{m_{\mathrm{s}} \cdot N}{2^{\lambda-1}}$$

(with appropriate changes to the runtime $\tilde{t}$).

3.2.2 Reducing to the DDH problem

Lemma 3.11 *Recall the assumptions, variables and events from the statement and proof of Theorem 3.3. In particular, assume that the DDH problem is (t, ε)-hard in $\mathbb{G}$. 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*

$$\Pr[Q_{\mathrm{s}} \wedge F_{\mathrm{DH}}] \leq N \cdot \varepsilon + \frac{m_{\mathrm{DH}} \cdot N^2}{|\mathbb{G}|}.$$

Intuition We will bound the simpler event F_{DH}. This event tells us that there is some safe node a in the GSD graph with encryption edges to nodes $u_1, \ldots, u_d$, where the query encrypt(a, u_i) returned the ciphertext $\langle g^{y_i}, \mathrm{Enc}_{k_i}(s_{u_i})\rangle$ with $k_i = H_{\mathrm{DH}}(g^{sk_a \cdot y_i})$, such that $g^{sk_a \cdot y_j}$ was the first hidden Diffie-Hellman key queried by $\mathcal{A}$ for some j. Moreover, at the instance $g^{sk_a \cdot y_j}$ was queried, no hidden seed had yet been queried by $\mathcal{A}$, implying that $\mathcal{A}$ had not queried $H_{\mathrm{gen}}(s_a)$ and thus had no information about sk_a (recall that $(pk_a, sk_a) = \mathrm{Gen}(H_{\mathrm{gen}}(s_a))$). It is interesting to note that our approach does not require that $\mathcal{A}$ has not queried H_{dep} for a hidden seed (i.e. that Q_{dep} was not triggered) as is implied by the event F_{DH}, because knowing $H_{\mathrm{gen}}(s_a)$ is the only way to learn about sk_a. Regardless, we still want to have our definition of F_{DH} include this information, as the bound on $\Pr\left[Q_{\mathrm{s}} \wedge \overline{F_{\mathrm{DH}}}\,\right]$ in Lemma 3.7 on page 14 relies on the fact that in the event of $Q_{\mathrm{s}} \wedge \overline{F_{\mathrm{DH}}}$ happening, Q_{DH} was not yet triggered when the event Q_{s} was triggered, i.e. when either the event Q_{gen} *or* the event Q_{dep} was triggered.

The intuition is clear that this means that $\mathcal{A}$ solved the Diffie-Hellman challenge (g^{sk_a}, g^{y_j}). What is not immediately clear is how to embed a *given* Diffie-Hellman challenge (g^x, g^y) from an instance of the DDH game and use $\mathcal{A}$ to tell whether the key k chosen by the challenger is the real key $g^{x \cdot y}$ or a uniformly random group element. An intuitive strategy would be to embed the challenge by setting $pk_a = g^x$ and $g^{y_j} = g^y$, which involves guessing u_j, and simply checking whether for any of the queries q_i to H_{DH} by $\mathcal{A}$ it holds that $q_i = k$. Now:

- If $k = g^{x \cdot y}$, $\mathcal{A}$ triggers F_{DH} and we guessed a and u_j correctly, then indeed as described above $q_i = g^{sk_a \cdot y_j} = k$ will hold for some i.
- If k is a random group element, then $\mathcal{A}$ has negligible probability of querying k, as no information about k is ever leaked to $\mathcal{A}$.

If we make sure not to change $\mathcal{A}$'s view of the game in the case $k = g^{x \cdot y}$ in this process, we can achieve an advantage of about $\Pr[F_{\mathrm{DH}}]/N^2$, where one factor $1/N$ arises from guessing a and another from guessing u_j. Unfortunately, this would yield no improvement over the result from [1].

Q: How to clarify that this was not my idea? We can avoid guessing u_j by being more clever about how we embed g^y. Instead of embedding g^y into a single encryption edge, we embed it into all encryption edges. To get a uniformly random exponent from y we set $y_j = y + r_j \mod |\mathbb{G}|$ with $r_j \leftarrow [|\mathbb{G}|]$. Given $g^{x \cdot y_j}$, we can easily compute $g^{x \cdot y}$:

$$g^{x \cdot y_j} = g^{x \cdot (y + r_j)} = g^{x \cdot y} \cdot g^{x \cdot r_j} \iff g^{x \cdot y} = g^{x \cdot y_j} \cdot \underbrace{((g^x)^{r_j})^{-1}}_{=: R_j}.$$

Now to determine whether k is the real Diffie-Hellman key, we check whether $q_i \cdot R_j = k$ for some i, j. This yields an advantage of about $\Pr[F_{\mathrm{DH}}]/N$ (and a slightly larger runtime). We can now proceed with the full proof.

Proof (of Lemma 3.11) As outlined above we use $\mathcal{A}$ to construct a DDH adversary $\mathcal{A}'$.

1. $\mathcal{A}'$ gets h_1, h_2 and k from the DDH challenger.
2. $\mathcal{A}'$ runs $\mathcal{A}$ to get n and D, samples $a \leftarrow [n]$ and initializes the GSD graph, seeds and the set of edges and corrupted nodes as in step 1 of the SD-GSD game, with the sole exception that $pk_a = h_1$ (as opposed to setting it to the public key output from $\mathrm{Gen}(H_{\mathrm{gen}}(s_a))$).
3. $\mathcal{A}'$ faithfully simulates the SD-GSD game to $\mathcal{A}$ with the following exception: For the j-th query $\mathrm{encrypt}(a, u_j)$ made by $\mathcal{A}$, $\mathcal{A}'$ replies with $\langle h_2 \cdot g^{r_j}, \mathrm{Enc}_{k_j}(s_{u_j}) \rangle$ where $r_j \leftarrow [|\mathbb{G}|]$, $k_j \leftarrow \{0,1\}^\kappa$. $\mathcal{A}'$ also computes and stores $R_j = \left(pk_a^{r_j}\right)^{-1}$.

   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 on an encryption edge $(u, v) \in E$ with $u \neq a$
     - $\mathcal{A}$ queries H_{gen} or H_{dep} for a hidden seed
     - $\mathcal{A}$ queries $\mathrm{corrupt}(u)$ for some node u such that a is no longer safe
   - If $\mathcal{A}$ queries q_i to H_{DH} such that $q_i \cdot R_j = k$ for some j, $\mathcal{A}'$ 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 (step 4 of the SD-GSD game), then $\mathcal{A}'$ outputs 0.

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

$$\mathrm{Adv}^{\mathrm{DDH}}_{(\mathbb{G},g)}(\mathcal{A}') = \left|\Pr\left[\mathcal{A}' \to 1 \mid b = 0\right] - \Pr\left[\mathcal{A}' \to 1 \mid b = 1\right]\right|, \tag{3.14}$$

where b is the bit sampled by the DDH challenger.

First, we will show that

$$\Pr\left[\mathcal{A}' \to 1 \mid b = 0\right] \geq \frac{\Pr[F_{\mathrm{DH}}]}{N}. \tag{3.15}$$

This part of the proof proceeds very similarly to the proof of Lemma 3.7 on page 14 and we will be a bit more concise. We focus on executions of $\mathrm{Game}^{\mathrm{DDH}}_{\mathcal{A},(\mathbb{G},g)}$ with $b = 0$. Let the games G_1, G_2 be defined as in Lemma 3.7, where we denote the node sampled at the beginning of each game by a_1, a_2, respectively (as opposed to w_1, w_2). Let $E = F_{\mathrm{DH}}$ and let E_1, E_2 and E' be the analogous events in G_1, G_2 and the game simulated by $\mathcal{A}'$ (note that in this latter game, the group elements $pk_a^{\log_g(h_2)+r_j}$ are also hidden Diffie-Hellman keys). Finally, we introduce the random variable

$$A = \begin{cases} 0 & \overline{F_{\mathrm{DH}}} \\ x & F_{\mathrm{DH}} \text{ holds and } Q_{\mathrm{DH}} \text{ was triggered on an encryption edge with source } x \end{cases}$$

(if x is not unique we choose the node with smallest identifier) and let A_1, A_2 and A' denote the corresponding random variables in game G_1, game G_2 and the game simulated by $\mathcal{A}'$.

Just as argued in Lemma 3.7,

$$\Pr[E_1] = \Pr[E] \tag{3.16}$$

holds, since whenever G_1 aborts, it is already decided whether F_{DH} holds:

- If the game was aborted when $\mathcal{A}$ queried a hidden Diffie-Hellman key, then F_{DH} holds.
- If the game was aborted when $\mathcal{A}$ queried H_{gen} or H_{dep} for a hidden seed, F_{DH} does not hold.

Next, the inequality

$$\Pr[A_1 = a_1 \mid E_1] \geq \frac{1}{N}$$

and therefore also

$$\Pr[A_1 = a_1] \geq \frac{1}{N} \cdot \Pr[E] \tag{3.17}$$

hold for the same reason that

$$\Pr[W_1 = w_1 \mid E_1] \geq \frac{1}{N}$$

and (3.7) held in Lemma 3.7.

Then, the equality

$$\Pr[A_1 = a_1] = \Pr[A_2 = a_2] \tag{3.18}$$

holds again due to the fact that when G_2 aborts because a_2 is no longer safe, we know that $A_2 \neq a_2$.

Finally, we need to argue that

$$\Pr[A_2 = a_2] = \Pr[A' = a]. \tag{3.19}$$

Consider how G_2 differs from the game simulated by $\mathcal{A}'$. As in Lemma 3.7, both games abort at exactly the same events (note that if $q_i \cdot R_j = k$ holds and $\mathcal{A}$ outputs 1, then $q_i = k \cdot R_j^{-1} = k \cdot pk_a^{r_j} = h_1^{\log_g(h_2)} \cdot pk_a^{r_j} = pk_a^{\log_g(h_2)+r_j}$, a hidden Diffie-Hellman key). The game simulated by $\mathcal{A}'$ differs in two aspects:

(i) $\mathcal{A}'$ sets $pk_a = h_1$ and not to the public key output by $\mathrm{Gen}(H_{\mathrm{gen}}(s_a))$

(ii) $\mathcal{A}'$ answers queries $\mathrm{encrypt}(a, u)$ differently

Note that as long as the game G_2 is ongoing, $\mathcal{A}$ has not queried H_{gen} for s_a or H_{DH} for a hidden Diffie-Hellman key. Both differences are therefore indistinguishable:

(i) By assumption the stated in ... (**TODO:** state assumption), running $\mathrm{Gen}(r)$ on a random bit string $r \leftarrow \{0,1\}^\lambda$ follows the same distribution as running $\mathrm{Gen}()$. The former process is behind the distribution of pk_a as viewed from $\mathcal{A}$ in G_2, as $\mathcal{A}$ has not queried $H_{\mathrm{gen}}(s_a)$, and the latter process is behind the distribution of pk_a in the game simulated by $\mathcal{A}'$, as the DDH challenger generates a public key with the same distribution as $\mathrm{Gen}()$. Since both processes follow the same distribution, pk_a follows the same in G_2 and the game simulated by $\mathcal{A}'$ from $\mathcal{A}$'s perspecive.

(ii) In G_2 a query $\mathrm{encrypt}(a, u)$ is answered with $\langle g^z, c\rangle$ where $z \leftarrow [|\mathbb{G}|], c \leftarrow \Pi_s.\mathrm{Enc}_k(s_u)$ and $k = H_{\mathrm{DH}}(pk_a^z)$. $\mathcal{A}'$ answers such a query with $\langle g^{\log_g(h_1)+r}, c'\rangle$ where $r \leftarrow [|\mathbb{G}|], c' \leftarrow \Pi_s.\mathrm{Enc}_{k'}(s_u)$ and $k' \leftarrow \{0,1\}^\kappa$. First, $\log_g(h_1) + r$ follows the same distribution as z. Second, pk_a^z is a hidden Diffie-Hellman and from $\mathcal{A}$'s view k follows the same distribution as k'.

Thus (3.19) indeed holds.

Now, again analogous to Lemma 3.7 if the event $A' = a$ occurred, then $\mathcal{A}'$ outputs 1 and

$$\begin{aligned}
\Pr[\mathcal{A}' \to 1 \mid b = 0] &\geq \Pr[A' = a] \\
&\overset{(3.19)}{=} \Pr[A_2 = a_2] \\
&\overset{(3.18)}{=} \Pr[A_1 = a_1] \\
&\overset{(3.17)}{\geq} \frac{\Pr[E]}{N} \\
&= \frac{\Pr[F_{\mathrm{DH}}]}{N},
\end{aligned}$$

Second, we will show that $\Pr[\mathcal{A}' \to 1 \mid b = 0]$ is negligible. When $b = 1$ in $\mathrm{Game}^{\mathrm{DDH}}_{\mathcal{A},(\mathbb{G},g)}$, k is a uniformly random group element independent of any information given to $\mathcal{A}$, in particular of $q_i \cdot R_j$ for any i, j. Thus for any i, j,

$$\Pr\left[q_i \cdot R_j = k\right] = \frac{1}{|\mathbb{G}|}.$$

Thus, by a union bound and using that $i \in [m_{\mathrm{DH}}], 1 \leq j \leq N - 1 \leq N$ (j is bounded by the maximum out-degree) we have

$$\Pr\left[\mathcal{A}' \to 1 \mid b = 0\right] \leq \frac{m_{\mathrm{DH}} \cdot N}{|\mathbb{G}|}. \tag{3.20}$$

Combining (3.14), (3.15) and (3.20) we get

$$\mathrm{Adv}^{\mathrm{DDH}}_{(\mathbb{G},g)}(\mathcal{A}') \geq \frac{\Pr[F_{\mathrm{DH}}]}{N} - \frac{m_{\mathrm{DH}} \cdot N}{|\mathbb{G}|}. \tag{3.21}$$

Furthemore, going through the details yields that $\mathcal{A}'$ runs in time

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

(the simulation of the SD-GSD game dominating the additional runtime). Then using the definition of $\tilde{t}$, with appropriately chosen constants we have

$$t_{\mathcal{A}'} \leq t.$$

So by virtue of the DDH problem being (t, ε)-hard in $\mathbb{G}$

$$\mathrm{Adv}^{\mathrm{DDH}}_{(\mathbb{G},g)}(\mathcal{A}') \leq \varepsilon$$

and if we combine this with (3.21) we get

$$\begin{aligned} \frac{\Pr[F_{\mathrm{DH}}]}{N} - \frac{m_{\mathrm{DH}} \cdot N}{|\mathbb{G}|} \leq \varepsilon \\ \iff \\ \Pr[F_{\mathrm{DH}}] \leq N \cdot \varepsilon + \frac{m_{\mathrm{DH}} \cdot N^2}{|\mathbb{G}|}, \end{aligned}$$

concluding the proof. ☐

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] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography, Third Edition*. Chapman & Hall/CRC, 3rd edition, 2020.

[7] 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.