Modular Design and Analysis Framework for Multi-Factor Authentication and Key Exchange

Nils Fleischhacker[1], Mark Manulis[1,2], and Amir Sadr-Azodi[1]

[1] Technische Universität Darmstadt, Germany
[2] University of Surrey, UK

Abstract. Multi-Factor Authentication (MFA), often coupled with Key Exchange (KE), offers very strong protection for secure communication and has been recommended by many major governmental and industrial bodies for the use in highly sensitive applications. Instantiations of the MFA concept vary in practice and in the research literature and various efforts in designing secure MFA protocols were unsuccessful.
This paper introduces a modular approach to the design and analysis of arbitrary MFAKE protocols, in form of an (α, β, γ)-MFAKE framework, that can accommodate multiple types and quantities of authentication factors, focusing on the three widely adopted categories that provide evidence of knowledge, possession, and physical presence. The framework comes with (i) a model for *generalized MFAKE* that implies many known flavors of single- and multi-factor Authenticated Key Exchange (AKE), and (ii) offers generic and modular constructions of secure MFAKE protocols that can be tailored to the needs of a particular application.
Our generic (α, β, γ)-MFAKE protocol is based on the new notion of *tag-based MFA* that in turn implies tag-based versions of many existing single-factor authentication schemes. We show examples and discuss generic ways to obtain tag-based flavors of password-based, public key-based, and biometric-based authentication protocols. By combining various tag-based single-factor authentication-only protocols, whose executions can be parallelized, with a single run of an Unauthenticated Key Exchange (UKE) we construct (α, β, γ)-MFAKE that is superior to a naïve black-box combination of multiple single-factor AKE schemes.

1 Introduction

Authentication Factors An *authentication factor* is used to produce some evidence that an entity at the end of the communication channel is the one which it claims to be. Modern computer security knows different types of authentication factors, all of which are widely used in practice. Their standard classification considers three main groups (see e.g. [18]), characterized by the nature of provided evidence: knowledge, possession, and physical presence. The evidence of knowledge is typically offered by low-entropy *passwords*. These include memorizable (long-term) passwords or PINs, e.g. for login purposes and banking ATMs, and one-time passwords that are common to many online banking and e-commerce transactions. The evidence of possession corresponds to physical devices such as smart cards, tokens, or TPMs, equipped with long-term (high-entropy) *secret keys* and some cryptographic functionality. These devices have tamper-resistance to protect secret keys from exposure. The evidence of physical presence refers to unique *biometric identifiers* of human beings. These may include face profile, finger prints, iris images, or behavioral characteristics such as gait or voice, and are used in access control systems and many electronic passports.

A different approach might be needed for an attacker to compromise a particular factor, depending on its type and use. For instance, passwords are susceptible to social engineering (e.g. phishing) and dictionary attacks. Digital devices can be lost or stolen. Those offering tamper-resistance may nonetheless be susceptible to reverse-engineering [19,20], side-channel attacks [17], and trojans (e.g. recent Sykipot Trojan attacks against smart cards). Biometric data can be obtained from a physical contact with the human being or copied if available in a digital form. Since the number of personal biometrics that admit efficient use in security technologies is limited, their wide use across different application domains makes it even harder to keep those factors private.

Multi-Factor Authentication (with Key Exchange) The strength of Multi-Factor Authentication (MFA) is based on the assumption that if an entity has many authentication factors, regardless of their nature, then it is hard for the attacker to compromise them all. That is, by combining different factors within a single authentication process, MFA aims at higher assurance in comparison to single-factor schemes. MFA has found its way into practice[3], most notable are combinations of long-term passwords with secret keys, possibly stored in tokens (e.g. Two-Factor SSH with USB sticks) or any of these with one-time passwords (e.g. OATH HOTP/TOTP, RSA SecurID, Google Authenticator). MFA is mostly used to authenticate a client/user to a remote server. As such, MFA authentication of the client is the main security goal. The server-side authentication in MFA protocols may offer additional protection, typically without using multiple factors on the server side.

The concept of Multi-Factor Authenticated Key Exchange (MFAKE), formalized in [32], extends MFA with establishment of secure session keys. In addition to authentication goals it aims at key secrecy, usually modeled in terms of (Bellare-Rogaway style) AKE-security [7,13]. Earlier MFAKE protocols focused mostly on two factors and were often unsuccessful: for instance, password-token combination from [30] was broken in [36] which itself was broken in [27], the scheme from [33] was cryptanalyzed in [35], and a biometric-token combination from [28] has fallen in [29]. Partially, these attacks were due to the missing modeling and analysis in those works.

A formal approach to MFAKE introduced in [32] was the first to account simultaneously for all three types of authentication factors. Most notable is their modeling of biometric factors. Unlike some previous single-factor biometric schemes, e.g. [16,9], that regarded biometrics as low- or high-entropy secrets, [32] drops secrecy in favor of *liveness assumption* (see also [12,11]) aiming at physical presence of a user. This approach was recently criticized in [24] where attacks against the MFAKE protocol from [32] were presented, allowing an adversary that steals user's password and impersonates the server to essentially compromise all other factors. These attacks stem, however, from the unfortunate design of the protocol in [32] and the way it combines the three factors; the attacks do not impose a generic threat against the liveness assumption (see Section 2.2 and Remark 1 for more discussion regarding the assumption). Nonetheless, after this break, design of an appropriate MFAKE protocol was left an open problem.

MFAKE protocols differ not only in nature of factors but also in their quantity. To this end, [34] introduced Multi-Factor Password AKE (MFPAKE), extending the PAKE setting [6], where arbitrary many low-entropy passwords (long-term and one-time) can be combined to authenticate the client. Their protocol offers server-side authentication and supports asymmetric PAKE setting from [21,22] in which servers store blinded passwords.

Generalized and Modular MFAKE Approach? Various problems in the design of secure MFAKE protocols, coupled with the fact that existing protocols differ in nature and quantity of deployed factors and that perception of MFA varies across products, standards, and research literature, motivates the need for a generalized and modular MFAKE approach. Aiming to build secure MFAKE protocols out of well-known and understood concepts behind existing single-factor solutions may help to avoid caveats, arising in the combination of factors and result in a cleaner design. The generality of the approach would help to understand relationships amongst MFAKE and other authentication schemes. Its modularity would offer opportunities for subsequent accommodation of new factors, e.g. for social authentication [10,15].

A naïve way to build general MFAKE is to look for different types of existing AKE protocols and try to combine them into a secure MFAKE solution. Although feasible (as also follows from our work), such approach is rather sub-optimal as combining different black-box AKE schemes would involve a lot of redundancy (due to the costs for establishing forward-secure keys) and reflect on the efficiency of the generic solution. In our work we take a slightly different approach, aiming to reduce redundancy, yet still enable modular and black-box design of arbitrary MFAKE protocols.

[3] MFA concept and its usage in practice are not consistently defined. For example, according to [1, Sec. 8.3], for two-factor authentication it suffices to deploy RADIUS authentication or use a single tamper-proof hardware token or a VPN access with individual certificate, whereas using two factors of the same type (e.g. two passwords) is not regarded as a two-factor solution. [2, Level 3] explicitly requires hardware tokens and some additional factor, e.g. password or biometric. This is more in line with the perception of MFA in research, where e.g. individual certificate alone is single-factor [32] but deployment of two or more passwords is multi-factor [34]. Aiming at generality, in this work we regard any scheme that deploys at least two factors, irrespective of their type, as an MFA solution.

1.1 Contributions and Organization

Generalized MFAKE Model and Relations We introduce and model the generalized notion of (α, β, γ)-MFAKE, including its MFA-only version, building on the three-factor model from [32]. In a classical client-server setting we admit arbitrary *quantities* and *combinations* of low-entropy passwords (long-term and one-time, symmetric and asymmetric), high-entropy secret keys (with the corresponding public keys), and biometric factors (with explicit and implicit matching). We model dictionary attacks on passwords and also account for the imperfect matching process of biometric templates. When modeling biometrics we follow the liveness assumption and do not treat biometric distributions as secret. We discuss why this approach seems more suitable in practice, and show that it admits a single model for all types of biometrics, which is not necessarily the case for the alternative approach that keeps biometrics private.

We shed light on the relationship of (α, β, γ)-MFAKE to existing concepts — by varying the parameters we show that all major existing single-factor and multi-factor settings can be seen as special cases of (α, β, γ)-MFAKE: these include (symmetric and asymmetric) PAKE [6,21] as $(1, 0, 0)$-MFAKE, two-party AKE protocols in the public key setting [7,13] through $(0, 1, 0)$-MFAKE, the notion of Biometric-based AKE (BAKE)[9,12] through $(0, 0, 1)$-MFAKE, as well as MFAKE from [32] through $(1, 1, 1)$-MFAKE and MFPAKE from [34] as $(\alpha, 0, 0)$-MFAKE.

Modular $(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma})$-MFAKE Framework We construct a generic (α, β, γ)-MFAKE protocol in a secure and modular way, based on simpler sub-protocols that can be instantiated from a wide range of existing, well-understood and efficient authentication-only schemes. More precisely we show how arbitrary many independent runs of authentication-only protocols based on passwords, secrets keys, and biometrics can be linked to a single independent session of an Unauthenticated Key Exchange (UKE) in a way that generically binds authentication and key establishment and results in an AKE-secure MFAKE protocol (with forward secrecy) that offers MFA for the client and strong (optional) authentication of the server.

To this end, we define a generalized notion of *tag-based* MFA, extending the preliminary concepts from [25] that considered the use of tags (auxiliary strings) in public key-based challenge-response scenarios. For all types of single-factor authentication-only protocols we demonstrate existence of efficient tag-based flavors and discuss their generic extensions with tags. We show how to use tags in (α, β, γ)-MFAKE protocol to bind all independent (black-box) sub-protocols in a secure way. (In this way, for example, we avoid the type of problems identified in [24] for the protocol in [32].) Our generic and modular MFAKE framework admits an MFA-compiler. We also discuss its generic optimizations.

ORGANIZATION. Generalized (α, β, γ)-MFAKE, its MFA-only version, and security goals are modeled in Section 2. Relationships to existing authentication approaches are presented in Section 3. Our modular and generic construction of (α, β, γ)-MFAKE is explained and analyzed in Section 4, along with the underlying tag-based MFA concept and its instantiations.

2 $(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma})$-MFAKE: Definitions and Security

We start with definitions of generalized MFAKE and its security model, presented as an extension of [32], which in turn builds on the classical Bellare-Rogaway approach [7,6].

2.1 System Model and Correctness

Participants, Sessions, and Partnering An MFAKE protocol is executed between two participants. A participant is either a *client* C or a *server* S. Several instances of any participant can exist at a time. This models multiple concurrent protocol sessions. An instance of participant $\mathrm{U} \in \{\mathrm{C}, \mathrm{S}\}$ in session s is denoted as $[\mathrm{U}, s]$. The session id s is the transcript of all messages sent and received by the instance, except for the very last protocol message. At the end of the protocol each instance either accepts or rejects.

By $\mathsf{pid}([\mathrm{U}, s])$ we denote *partner identity* with which $[\mathrm{U}, s]$ believes to be interacting in the protocol session. Two instances $[\mathrm{U}, s]$ and $[\mathrm{U}', s']$ are said to be *partnered* if and only if $\mathsf{pid}([\mathrm{U}, s]) = \mathrm{U}'$, $\mathsf{pid}([\mathrm{U}', s']) = \mathrm{U}$, and their session ids match [7], denoted $s = s'$.

Authentication Factors Each client may have arbitrary types and quantities of authentication factors that it may use in multiple protocol sessions as detailed in the following.

PASSWORDS A client C may hold an array $\boldsymbol{pwd}_{\mathrm{C}}$ of α *passwords* $\boldsymbol{pwd}_{\mathrm{C}}[i]$, $i = 1, \ldots, \alpha$. Each password $\boldsymbol{pwd}_{\mathrm{C}}[i]$ is assumed to have low-entropy, chosen from some dictionary $\mathcal{D}_{\mathtt{pwd}}$. In general, passwords can be used across multiple protocol sessions, in which case they are considered to be long-term. We also admit one-time passwords [3,31] that have been previously registered with the server as well as an asymmetric setting, e.g. [21,22], where the server stores some non-trivial function of a password — if the server gets compromised such asymmetry usually implies that an offline dictionary attack is still necessary to impersonate the client.

CLIENT SECRET KEYS A client C may hold an array $\boldsymbol{sk}_{\mathrm{C}}$, containing β *secret keys* $\boldsymbol{sk}_{\mathrm{C}}[i] \in \mathtt{KeySp}$, $i = 1, \ldots, \beta$, e.g. signing keys. The array of corresponding public keys is denoted $\boldsymbol{pk}_{\mathrm{C}}$ and is assumed to be known system-wide. It is assumed that all secret keys of the client are independent and have high entropy, i.e., $\Pr[sk \in \boldsymbol{sk} | sk \leftarrow_R \mathtt{KeySp}] = \frac{\beta}{|\mathtt{KeySp}|}$ is assumed to be negligible. Note that if any of the private keys in $\boldsymbol{sk}_{\mathrm{C}}$ is stored in a secure device (e.g. in a smartcard or TPM) then its usage in the protocol essentially implies client's access to the corresponding device, i.e., the model doesn't distinguish in this case between the devices and private keys of the client.

BIOMETRICS For each client C there are γ public biometric distributions $\mathtt{Dist}_{\mathrm{C},i}$, $i = 1, \ldots, \gamma$. The process of measuring some biometric (being it face, any particular finger, or iris) is comprehended as drawing a *biometric template* $W_{\mathrm{C},i}$ according to $\mathtt{Dist}_{\mathrm{C},i}$. Upon the enrollment of the client an array $\boldsymbol{W}_{\mathrm{C}}$ containing γ biometric templates $\boldsymbol{W}_{\mathrm{C}}[i]$, $i = 1, \ldots, \gamma$ is created and will be used as a reference for the session-dependent matching process on the server's side. We do not require that $\boldsymbol{W}_{\mathrm{C}}$ is stored in clear on the server's side. That is, our model admits the case where the server stores some non-trivial transformation of $\boldsymbol{W}_{\mathrm{C}}$, e.g. based on secure sketches [16,9].

Functionality of biometric data matching is modeled through an algorithm $\mathsf{BioMatch}$, which takes as input a candidate template W^* and a reference template W, which may also be given *implicitly* in a transformed form, and outputs 1 indicating that W^* matches W and 0 otherwise. For example, $\mathsf{BioMatch}$ can require that the Hamming distance between W and W^* remains below some threshold, an approach used, e.g., in [9,32]. We also take into account that biometric measurements are *not* perfect:

– For any client C,

$$\Pr\left[\mathsf{BioMatch}(W^*_{\mathrm{C},i}, \boldsymbol{W}_{\mathrm{C}}[i]) = 1 \mid W^*_{\mathrm{C},i} \leftarrow \mathtt{Dist}_{\mathrm{C},i}, i \in [1, \gamma]\right] \geq 1 - \mathsf{false}^{\mathrm{neg}}_i,$$

where $\mathsf{false}^{\mathrm{neg}}_i$ is the probability with which ith biometric of C is *falsely rejected.*

– For any two clients C′, C with $\mathrm{C}' \neq \mathrm{C}$,

$$\Pr\left[\mathsf{BioMatch}(W^*_{\mathrm{C}',i}, \boldsymbol{W}_{\mathrm{C}}[i]) = 0 \mid W^*_{\mathrm{C}',i} \leftarrow \mathtt{Dist}_{\mathrm{C}',i}, i \in [1, \gamma]\right] \geq 1 - \mathsf{false}^{\mathrm{pos}}_i$$

where $\mathsf{false}^{\mathrm{pos}}_i$ is the probability with which ith biometric of C′ is *falsely accepted.*

While false rejection is relevant for MFA correctness, false acceptance impacts the lower bounds of the protocol's security.

SERVER SECRET KEY We assume that server S may have a high-entropy secret key sk_{S} with the corresponding system-wide known public key pk_{S}.

Generalized Multi-Factor Authenticated Key Exchange We define generalized MFAKE and its correctness property.

Definition 1 ((α, β, γ)-Multi-Factor Authenticated Key Exchange). *A generalized Multi-Factor Authenticated Key Exchange* (α, β, γ)-$\mathsf{MFAKE}(\mathrm{C}, \mathrm{S})$ *is a two-party protocol, executed between a client instance* $[\mathrm{C}, s]$ *with* α *passwords,* β *secret keys, and* γ *biometric templates, and a server instance* $[\mathrm{S}, s']$ *such that at the end of their interaction each instance either accepts or rejects. The correctness property of the protocol requires that for all* $\kappa \in \mathbb{N}$, *if at the end of the protocol session* $[\mathrm{C}, s]$ *accepts holding session key* k_{C} *and* $[\mathrm{S}, s]$ *accepts holding session key* k_{S}, *and* $[\mathrm{C}, s]$ *and* $[\mathrm{S}, s]$ *are partnered, then* $\Pr[k_{\mathrm{C}} = k_{\mathrm{S}}] = 1$.

Generalized Multi-Factor Authentication In MFA protocols parties either reject or accept their communication partner without computing a session keys. The following definition for multi-factor authentication of the client towards the server accounts for imperfect biometric matching process where servers may falsely reject valid clients.

Definition 2 **((α, β, γ)-Multi-Factor Authentication).** *A generalized multi-factor authentication protocol* (α, β, γ)-MFA *is a two-party protocol, executed between a client instance* $[\mathrm{C}, s]$ *with* α *passwords,* β *secret keys, and* γ *biometric templates, and a server instance* $[\mathrm{S}, s']$ *such that at the end of their interaction the server instance either accepts* C *as its communication partner or rejects. Let '*$\mathtt{acc}\ \mathrm{C}$*' denote the event that* $[\mathrm{S}, s]$ *accepts the client. The correctness property of the* (α, β, γ)-MFA *protocol requires that* $\Pr[\mathtt{acc}\ \mathrm{C}] \geq 1 - \sum_{i=1}^{\gamma} \mathsf{false}_i^{\mathrm{neg}}$.

For optional *server-side authentication* the multi-factor aspect is typically irrelevant, i.e. the client instance either accepts the server based on its public key pk_{S} or rejects it. The correctness property in this case is perfect.

2.2 Security Model: AKE-Security and Mutual Authentication

MFAKE protocols aim to guarantee standard goals with respect to session key security and authentication, against any probabilistic polynomial-time adversary $\mathcal{A}$. Considering the asymmetry with regard to the use of multiple authentication factors on the client side and typically one authentication factor on the server's side, modeling of these security goals for MFAKE protocols proceeds slightly different than in protocols with a single factor on both sides.

Liveness Assumption for Biometrics We assume that biometric data is public and resort to liveness assumption [32] to ensure physical presence of a user. Liveness of a client C is modeled through a special *biometric computation oracle* $\mathsf{BioComp}([\mathrm{C}, s], W_{\mathcal{A},i})$: depending on the state of $[\mathrm{C}, s]$ this oracle uses client's secret keys $\boldsymbol{sk}_{\mathrm{C}}$ and passwords $\boldsymbol{pwd}_{\mathrm{C}}$ together with an input biometric template $W_{\mathcal{A},i}$ that must be chosen according to some *adversary-specified* distribution $\mathtt{Dist}_{\mathcal{A},i}$ to perform the required computation step that would otherwise be performed using a template $W^*_{\mathrm{C},i}$ chosen according to the distribution $\mathtt{Dist}_{\mathrm{C},i}$. The crucial condition here is that $\mathtt{Dist}_{\mathcal{A},i}$ must significantly differ from $\mathtt{Dist}_{\mathrm{C},i}$ such that $\Pr[\mathsf{BioMatch}(W_{\mathcal{A},i}, \boldsymbol{W}_{\mathrm{C}}[i]) = 0 \mid W_{\mathcal{A},i} \leftarrow_R \mathtt{Dist}_{\mathcal{A},i}] \geq 1 - \mathsf{false}_i^{\mathrm{pos}}$. For simplicity, we assume that $\mathcal{A}$ queries $\mathsf{BioComp}$ only with templates $W_{\mathcal{A},i}$ from the distributions $\mathtt{Dist}_{\mathcal{A},i}$, $1 \leq i \leq \gamma$ (alternative modeling of $\mathsf{BioComp}$ would require $\mathcal{A}$ to specify some template generation algorithm with a suitable distribution $\mathtt{Dist}_{\mathcal{A},i}$ which will be invoked within $\mathsf{BioComp}$ on each new query to pick $W_{\mathcal{A},i}$). Liveness assumption requires that any *new* message m, whose computation depends on the ith biometric template of C, must be previously generated by the $\mathsf{BioComp}$ oracle, before an active adversary can make use of it. $\mathsf{BioComp}$ oracle allows the adversary to test own biometric templates in the computations of the client. Note that the liveness assumption allows for replay attacks on biometric-dependent messages, i.e. $\mathcal{A}$ can consult the $\mathsf{BioComp}$ oracle to obtain a new message in one session of C and then replay it in another.

Remark 1. Hao and Clarke [24] criticized the MFAKE model from [32] for the assumption that biometric data is public. They argued that biometric templates that can be obtained by the adversary in practice, like fingerprints from surfaces, are often of poor quality and that for this reason obtaining high-quality biometric templates should be seen as a corruption of the client. This might be a valid argument in certain use cases, however, for the purpose of generality, it seems more appropriate to assume that biometric data is public and resort to the liveness assumption, when modeling security of biometric-based protocols. Since biometric data is used in different domains (e.g. electronic passports, personal computers, entry access systems, etc.) leakage of high-quality templates is not unlikely. In contrast to private keys, biometric characteristics are produced by nature and are bound to a specific person. From this perspective, their modeling via liveness assumption, aiming at user's physical presence seems to be more to the point. Liveness assumption has also been in the focus of recent standardization initiatives, e.g. ISO/IEC WD 30107 Anti-Spoofing and Liveness Detection Techniques. We acknowledge that liveness assumption is strong but this would also apply to an assumption that biometrics are kept secret. We note that attacks in

[24] against the protocol in [32] are specific to the latter's design and the way in which it processes biometric data. These attacks do not discover a general weakness of protocols based on liveness assumption, and in particular, they do not apply to our modular protocol from Section 4.3.

Client and Server Corruptions The adversary $\mathcal{A}$ is active and may corrupt parties. Client corruptions are handled through the $\mathsf{CorruptClient}(\mathrm{C}, \mathtt{type}, i)$ oracle. The adversary can indicate which authentication factor (its type and position in the corresponding array) of C should be considered as corrupted. Corrupting passwords and secret keys reveals them to $\mathcal{A}$ whereas corruption of biometric factors implies that $\mathcal{A}$ needs no longer to follow restrictions put forth by the liveness assumption for those factors. $\mathcal{A}$ can ask multiple $\mathsf{CorruptClient}$ queries for different factors of its choice. This models realistic scenarios where different factors may require different attacks. Server corruptions are handled through the $\mathsf{CorruptServer}$ oracle which responds with sk_{S}.

Adversarial Queries Our security definitions will be given in form of games. $\mathcal{A}$ can interact with the instances through a set of oracles, as specified in the following. We assume that $\mathrm{U}, \mathrm{U}' \in \{\mathrm{C}, \mathrm{S}\}$.

$\mathsf{Invoke}(\mathrm{U}, \mathrm{U}')$ allows $\mathcal{A}$ to invoke a session at party U with party U$'$. If U is a client then U$'$ must be a server, and vice versa. In response, a new instance $[\mathrm{U}, s]$ with $\mathsf{pid}([\mathrm{U}, s]) = \mathrm{U}'$ is established. $[\mathrm{U}, s]$ takes as input the authentication factors of U. If $[\mathrm{U}, s]$ is supposed to send a message first then this message is generated and given to $\mathcal{A}$.

$\mathsf{Send}([\mathrm{U}, s], m)$ allows $\mathcal{A}$ to send messages to the protocol instances (e.g. by forwarding, modifying, or creating new messages). In general, the oracle processes m according to the state of $[\mathrm{U}, s]$ and eventually outputs the next message (if any) to $\mathcal{A}$. However, if $\mathrm{U} = \mathrm{S}$ and m is such that it was not produced by an instance of $\mathrm{C} = \mathsf{pid}([\mathrm{S}, s])$ but its computation was expected to involve ith biometric of C, then m is processed only if it was output by $\mathsf{BioComp}([\mathrm{C}, \cdot], W_{\mathcal{A},i})$ or if $\mathcal{A}$ previously queried $\mathsf{CorruptClient}(\mathrm{C}, 3, i)$.

$\mathsf{BioComp}([\mathrm{C}, s], W_{\mathcal{A},i})$ outputs message m (if any) computed based on the internal state of $[\mathrm{C}, s]$ using $\boldsymbol{sk}_{\mathrm{C}}$, $\boldsymbol{pwd}_{\mathrm{C}}$, and $W_{\mathcal{A},i}$ (from $\mathtt{Dist}_{\mathcal{A},i}$ as explained above).

$\mathsf{RevealSK}([\mathrm{U}, s])$ gives $\mathcal{A}$ the session key computed by $[\mathrm{U}, s]$ (if such key exists).

$\mathsf{CorruptClient}(\mathrm{C}, \mathtt{type}, i)$ allows $\mathcal{A}$ to corrupt authentication factors of C. If $\mathtt{type} = 1$ then $\mathcal{A}$ is given $\boldsymbol{pwd}_{\mathrm{C}}[i]$; if $\mathtt{type} = 2$ then it receives $\boldsymbol{sk}_{\mathrm{C}}[i]$; if $\mathtt{type} = 3$ then $\mathcal{A}$ receives nothing but the liveness assumption for the ith biometric of C is dropped.

$\mathsf{CorruptServer}(\mathrm{S})$ gives $\mathcal{A}$ the secret key sk_{S}.

Freshness The notion of freshness in key exchange models typically prevents $\mathcal{A}$ from using its oracles to attack the protocol in a trivial way. For instance, key secrecy and authentication goals will require that no protocol participant was fully corrupted during the protocol session: a *client* C *is fully corrupted* if and only if all existing authentication factors of C have been corrupted via corresponding $\mathsf{CorruptClient}(\mathrm{C}, \cdot, \cdot)$ queries; a *server* S *is fully corrupted* if and only if a $\mathsf{CorruptServer}(\mathrm{S})$ query has been asked.

Similar to [32], our definition of freshness aims at server instances since $\mathcal{A}$ will be required to break AKE-security for their session keys. This is not a limitation since correctness property guarantees that any accepted partnered client instance will compute the same key as the server instance. In protocols without server-side authentication $\mathcal{A}$ can execute the role of the server and compute the same key as the client.

An accepted instance $[\mathrm{S}, s]$ is said to be *fresh* if all of the following holds:

– Upon acceptance of $[\mathrm{S}, s]$ neither S nor $\mathrm{C} = \mathsf{pid}([\mathrm{S}, s])$ were fully corrupted.
– There has been no $\mathsf{RevealSK}$ query to $[\mathrm{S}, s]$ or to its partnered client instance (if such instance exists).

The above conditions allow full corruption of parties after the session ends (upon acceptance) and thus also model the property of *forward secrecy.*

Security of Session Keys Secrecy of session keys is modeled in terms of AKE-security in the Real-or-Random indistinguishability framework [4], which involves additional Test queries. These can be asked only to fresh instances $[\mathrm{S}, s]$. Their answer depends on the value of bit b, which is fixed in the beginning of the game: if $b = 1$ then $\mathcal{A}$ receives the real session key held by $[\mathrm{S}, s]$; if $b = 0$ then $\mathcal{A}$ is given a random key chosen uniformly from the set of all possible session keys. At the end of the game $\mathcal{A}$ outputs bit b' aiming to guess b. Let $\mathsf{Succ}_{\mathrm{AKE}}^{\mathcal{A},(\alpha,\beta,\gamma)\text{-}\mathsf{MFAKE}}(\kappa)$ denote the probability of the event $b' = b$ in a game played by $\mathcal{A}$ against the AKE-security of (α, β, γ)-MFAKE. Let q denote the total number of invoked sessions. (α, β, γ)-MFAKE is AKE-secure, if for all PPT adversaries $\mathcal{A}$ the following advantage is negligible in κ:

$$\mathsf{Adv}_{\mathrm{AKE}}^{(\alpha,\beta,\gamma)\text{-}\mathsf{MFAKE},\mathcal{A}}(\kappa) = \left|\mathsf{Succ}_{\mathrm{AKE}}^{(\alpha,\beta,\gamma)\text{-}\mathsf{MFAKE},\mathcal{A}}(\kappa) - q\left(\frac{\alpha}{|\mathcal{D}_{\mathtt{pwd}}|} + \sum_{i=1}^{\gamma} \mathsf{false}_i^{\mathrm{pos}}\right) - \frac{1}{2}\right|.$$

AKE-security is relevant only for (α, β, γ)-MFAKE protocols from Definition 1. It doesn't apply to (α, β, γ)-MFA protocols from Definition 2 that do not support key establishment.

Authentication Goals An (α, β, γ)-MFAKE protocol must further provide authentication. We separate this goal into two flavors, for the client and for the server. A protocol which provides both is said to offer *mutual authentication.*

Client Authentication. Let $\mathcal{A}$ be an adversary against client authentication of (α, β, γ)-MFAKE that interacts with client and server instances using the aforementioned queries (whereby Test queries are irrelevant). $\mathcal{A}$ breaks client authentication if there exists a server instance $[\mathrm{S}, s]$ that has accepted a client $\mathrm{C} = \mathsf{pid}([\mathrm{S}, s])$, for which there exists no client instance that is partnered with $[\mathrm{S}, s]$, and neither S nor C were fully corrupted upon the acceptance of $[\mathrm{S}, s]$.

Let $\mathsf{Succ}_{\mathrm{CAuth}}^{(\alpha,\beta,\gamma)\text{-}\mathsf{MFAKE},\mathcal{A}}(\kappa)$ denote the probability with which $\mathcal{A}$ breaks client authentication of (α, β, γ)-MFAKE. The protocol is said to be CAuth-secure, if for all PPT adversaries $\mathcal{A}$ the following advantage is negligible in the security parameter κ:

$$\mathsf{Adv}_{\mathrm{CAuth}}^{(\alpha,\beta,\gamma)\text{-}\mathsf{MFAKE},\mathcal{A}}(\kappa) = \left|\mathsf{Succ}_{\mathrm{CAuth}}^{(\alpha,\beta,\gamma)\text{-}\mathsf{MFAKE},\mathcal{A}}(\kappa) - q\left(\frac{\alpha}{|\mathcal{D}_{\mathtt{pwd}}|} + \sum_{i=1}^{\gamma} \mathsf{false}_i^{\mathrm{pos}}\right)\right|.$$

This definition of CAuth-security is directly applicable to (α, β, γ)-MFA protocols from Definition 2. The advantage of $\mathcal{A}$ is denoted then $\mathsf{Adv}_{\mathrm{CAuth}}^{(\alpha,\beta,\gamma)\text{-}\mathsf{MFA},\mathcal{A}}(\kappa)$ and its success probability is subject to the same bounds as $\mathsf{Succ}_{\mathrm{CAuth}}^{(\alpha,\beta,\gamma)\text{-}\mathsf{MFAKE},\mathcal{A}}(\kappa)$. For (α, β, γ)-MFA protocols CAuth-security is the main property.

Remark 2. Due to low-entropy of passwords and non-perfect biometric matching the success probability of $\mathcal{A}$ in breaking client authentication has a lower bound of $q(\frac{\alpha}{|\mathcal{D}_{\mathtt{pwd}}|} + \sum_{i=1}^{\gamma} \mathsf{false}_i^{\mathrm{pos}})$. This contrasts to server authentication defined in the next paragraph.

Server Authentication. When it comes to server authentication, the roles of the C and S are swapped. An adversary $\mathcal{A}$ against server authentication of (α, β, γ)-MFAKE interacts with client and server instances and breaks server authentication if there exists a client instance $[\mathrm{C}, s]$ that has accepted a server $\mathrm{S} = \mathsf{pid}([\mathrm{C}, s])$, for which there exists no server instance that is partnered with $[\mathrm{C}, s]$, and neither C nor S were fully corrupted upon the acceptance of $[\mathrm{C}, s]$. (α, β, γ)-MFAKE is SAuth-secure, if for all PPT adversaries $\mathcal{A}$ the probability of breaking server authentication, denoted $\mathsf{Succ}_{\mathrm{SAuth}}^{(\alpha,\beta,\gamma)\text{-}\mathsf{MFAKE},\mathcal{A}}(\kappa)$ is negligible in the security parameter κ.

3 Some Special Cases of (α, β, γ)-MFAKE

We demonstrate usefulness of generalized (α, β, γ)-MFAKE definitions by discussing their relationship with some existing notions.

$(1,0,0)$-MFAKE $=$ PAKE Our generalized MFAKE captures Password-based Authenticated Key Exchange (PAKE) as defined in [6] and later extended to address the asymmetric case in [21]. We simply omit optional server authentication and set $(\alpha,\beta,\gamma)=(1,0,0)$, meaning that both S and C share the same password. Our freshness condition will still guarantee that this password remains uncorrupted for fresh server instances. At first sight, there is a problem that our $(1,0,0)$-MFAKE model allows the adversary to test only server instances, while in the PAKE model Test queries can also be asked to the instances of the client. However, this is not a limitation since in the special case of $(1,0,0)$-MFAKE model the roles C and S are symmetric (as opposed to the general case where C may have more authentication factors than S). Hence, we can still apply our AKE-security definition to PAKE [6,21] by first proving AKE-security for the PAKE server and then for its client viewing the latter as S. Definition of client authentication in $(1,0,0)$-MFAKE already matches that in [6]. The additional definition of server authentication from [6] is obtained from our client authentication by viewing PAKE server as C.

$(0,1,0)$-MFAKE $=$ 2-AKE The standard notion of two-party Authenticated Key Exchange (AKE) [7,13], defines AKE-security in the public-key setting with respect to the protocol participants U and U′ and Test queries can be asked likewise to the instances of the both. We obtain this notion using $(\alpha,\beta,\gamma)=(0,1,0)$ and considering server authentication based on sk_{S} as a necessary requirement. Similar to the PAKE case above, our $(0,1,0)$-MFAKE model with server authentication provides symmetry for the roles C and S. Hence, proving AKE-security as defined in [7,13] can be achieved by proving AKE-security in our $(0,1,0)$-MFAKE model for two different cases $(\mathrm{C},\mathrm{S})=(\mathrm{U},\mathrm{U}')$ and $(\mathrm{C},\mathrm{S})=(\mathrm{U}',\mathrm{U})$. We remark that our (α,β,γ)-MFAKE model doesn't consider leakage of ephemeral secrets [26]. Resilience against these attacks aims to strengthen security of 2-AKE protocols whereas in MFAKE protocols strengthening comes from the use of multiple factors. Various models aiming at leakage of ephemeral secrets in 2-AKE protocols are incomparable and may not necessarily offer a stronger protection, as demonstrated by Cremers [14].

$(0,0,1)$-MFAKE $=$ BAKE By setting $(\alpha,\beta,\gamma)=(0,0,1)$ our definitions of AKE-security and client authentication capture the notion of Biometric-based Authenticated Key Exchange (BAKE) [9,12], in the presence of liveness assumption. Some BAKE protocols, e.g. [9,16], assume that biometric data is secret and adopt special techniques, e.g. secure sketches and robust fuzzy extractors [16,8] to let the client and server first derive identical secret high-entropy shared keys. With such pre-processing in place the resulting protocol can then be analyzed in the 2-AKE model from [7]. If, however, biometric template of the client is used in a "raw format" without pre-processing step then this model is not general enough as it doesn't consider imperfect correctness via false negatives and potential security degradation via false positives in the matching process. If secret biometric data has low entropy, which is not increased by the pre-processing step then instead of [7] one would have to resort to a PAKE model in order to account for offline dictionary attacks against such biometrics.

Our BAKE or $(0,0,1)$-MFAKE model with public distribution of biometric data is equally applicable to low- and high-entropy biometrics. Indeed, irrespective of their entropy the adversary would have to consult its BioComp oracle for each new biometric-dependent message, as long as the liveness assumption is in place for that biometric.

Simple $(0,0,1)$-MFAKE Protocol. As observed in [32], BAKE protocols with public biometric data have not been defined so far. The actual MFAKE construction in [32] doesn't admit a pure BAKE protocol since it binds biometric data to the password and it is not clear how to separate the both. In fact this link lead to an attack in [23]. We give now a simple construction of an AKE-secure $(0,0,1)$-MFAKE whose explicit matching process checks that the Hamming distance of a candidate template W'_{C} and the reference template W_{C} (stored at S)remains below a threshold τ.

Let $(\mathbb{G},g,q)$ be a cyclic group of sufficiently large prime order q. C and S first execute an unauthenticated Diffie-Hellman key exchange in $\mathbb{G}$ by exchanging g^x and g^y. Consider two hash functions $\mathcal{H}_1,\mathcal{H}_2:\mathbb{G}\mapsto\{0,1\}^\kappa$. Let $W'_{\mathrm{C},i}$ resp. $W_{\mathrm{C},i}$ denote the ith bit of the corresponding template. For each bit i the client computes $h_i=\mathcal{H}_1(g^x,g^y,g^{xy},W'_{\mathrm{C},i},i)$ using its version of g^{xy} and sends the resulting set $\{h_i\}_i$ to S. S re-computes corresponding values using its version of g^{xy}

and the reference template W_{C}, and accepts with $k_{\mathrm{S}} = \mathcal{H}_2(g^x, g^y, g^{xy})$ if strictly less than τ hash values from $\{h_i\}_i$ do not match. It is not difficult to see that this protocol guarantees AKE-security for the session keys k_{S} if liveness assumption is in place, which essentially prevents the adversary from sending any h_i that was not computed beforehand through the $\mathsf{BioComp}$ oracle. The secrecy of the session key k_S follows then from the classical CDH assumption in the random oracle model.

$(\alpha, 0, 0)$-MFAKE = MFPAKE [34] We obtain Multi-Factor Password Authenticated Key Exchange (MFPAKE) from [34] by setting β and γ to 0. In MFPAKE a client shares multiple passwords with the remote server. MFPAKE accounts for one-time passwords and for the asymmetric case, where passwords are blinded and stored on the server side. AKE-security definition from [34] considers forward secrecy and requires that session keys remain secure as long as one of the passwords remains uncorrupted. Our $(\alpha, 0, 0)$-MFAKE model captures this goal through the definition of AKE-security.

$(1, 1, 1)$-MFAKE = MFAKE [32] Since our (α, β, γ)-MFAKE model extends the model from [32] it is not surprising that we obtain their setting using $(\alpha, \beta, \gamma) = (1, 1, 1)$.

4 A Secure (α, β, γ)-MFAKE Protocol

Our generalized MFAKE protocol, denoted (α, β, γ)-MFAKE, is built in a modular way from sub-protocols for different authentication factors, yet with some extensions and optimizations. We start with its main building blocks.

4.1 Tag-based Authentication

Tag-based Authentication (TbA), introduced in [25], accounts for the use of auxiliary, possibly public, strings (tags) in authentication protocols. In TbA each party uses a tag, in addition to the authentication factor, and the protocol guarantees that if parties accept then their tags match. For instance, the server accepts some client in a session if and only if that client was alive during that session and used as input the same tag as the server. Considering a class of public key-based challenge response protocols, [25] gave a signature-based generic compiler for obtaining the TbA property. In our work we adopt a more general TbA notion defined in the following.

Definition 3 (Tag-based Multi-Factor Authentication). *A generalized tag-based MFA protocol (α, β, γ)-tMFA is an (α, β, γ)-MFA protocol from Definition 2, where in addition the client instance $[\mathrm{C}, s]$ takes as input tag t_{C}, the server instance $[\mathrm{S}, s]$ takes as input tag t_{S}, and if $t_{\mathrm{C}} \neq t_{\mathrm{S}}$ then both parties reject; otherwise, they accept as in the (α, β, γ)-MFA protocol.*

Tag-based CAuth-security: *Let $\mathcal{A}$ be a PPT adversary against client authentication of (α, β, γ)-tMFA that interacts with the instances of* C *and* S *using the same oracles as for (α, β, γ)-MFA, except that the* Invoke *oracle is modified such that it receives tag t as an additional input from $\mathcal{A}$. $\mathcal{A}$ is said to break* CAuth-security *of (α, β, γ)-tMFA if at the end of its interaction there exists a server instance $[\mathrm{S}, s]$ that was invoked with tag t_{S} and has accepted a client $\mathrm{C} = \mathsf{pid}([\mathrm{S}, s])$, for which there exists no client instance that was invoked with tag $t_{\mathrm{C}} = t_{\mathrm{S}}$ and is partnered with $[\mathrm{S}, s]$, and neither* S *nor* C *were fully corrupted upon the acceptance of $[\mathrm{S}, s]$. The corresponding advantage of $\mathcal{A}$, denoted $\mathsf{Adv}_{\mathrm{CAuth}}^{(\alpha,\beta,\gamma)\text{-}\mathsf{tMFA},\mathcal{A}}(\kappa)$, is then defined analog to the advantage in (α, β, γ)-MFA.*

$\mathcal{A}$ is allowed to test tags of its own choice, i.e. existence of a partnered client instance that was invoked with a tag $t_{\mathrm{C}} \neq t_{\mathrm{S}}$ leads to a successful attack. Definitions of *tag-based server authentication* in (α, β, γ)-tMFA and success probability $\mathsf{Succ}_{\mathrm{SAuth}}^{(\alpha,\beta,\gamma)\text{-}\mathsf{tMFA},\mathcal{A}}(\kappa)$ are obtained by swapping the roles of C and S, as for (α, β, γ)-MFA in Section 2.2.

4.2 Utilized Sub-Protocols and Their Examples

Our generic MFAKE protocol is built from sub-protocols that represent special cases of tag-based MFA. We first describe corresponding (non tag-based) authentication-only protocols and provide some examples, including the discussion on how to extend those protocols with tags.

PwA : (Tag-based) Password-based Authentication Protocol The first sub-protocol is for password-based authentication, denoted PwA, in which only one party (in our case the client) authenticates itself to the other party (server). In our generalized MFA model the adversarial advantage against client authentication of PwA becomes $\mathsf{Adv}^{\mathcal{A},\mathsf{PwA}}_{\mathrm{CAuth}}(\kappa) = \mathsf{Adv}^{\mathcal{A},(1,0,0)\text{-}\mathsf{MFA}}_{\mathrm{CAuth}}(\kappa)$.

In fact, any AKE-secure PAKE protocol with key confirmation from client to server can already be used as PwA. On the other hand, PAKE protocols can be simplified since we do not require PwA to provide session keys. The following is an example for the PAKE protocol from [5] when only client-side authentication with key confirmation is applied.

PwA EXAMPLE. Let $(\mathbb{G}, g, q)$ be a description of the cyclic group of prime order q with generator g that together with some element $V \in \mathbb{G}$ and a hash function $\mathcal{H} : \{0,1\}^* \mapsto \{0,1\}^\kappa$ build public parameters. Assume that $pwd \in \mathbb{Z}_q$ is shared between C and S. In a PwA session, derived from [5], S sends $Y = g^y$ for some $y \leftarrow_R \mathbb{Z}_q$ to C. C picks $x \leftarrow_R \mathbb{Z}_q$ and responds with $(X^*, h) = (g^x V^{pwd}, \mathcal{H}(\mathrm{C}, \mathrm{S}, Y, X^*, Y^x, pwd))$. S checks whether $h = \mathcal{H}(\mathrm{C}, \mathrm{S}, Y, X^*, (X^*/V^{pwd})^y, pwd)$ and accepts the client in this case. It is easy to see that client authentication of this PwA follows from the security of PAKE in [5].

ASYMMETRIC PwA. In [22] the authors introduce a generic transformation, called Ω-method (as a generalization of [21]), that converts any PAKE into an asymmetric PAKE where passwords are stored on the server side in a blinded way. Roughly, this method uses a random oracle $\mathcal{H}'$, a symmetric encryption scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, and an additional pair of signing keys (sk, pk), which are not treated as an authentication factor. For a given password pwd the server stores $(\mathcal{H}'(pwd)$, $\mathsf{Enc}_{pwd}(sk))$. The Ω-method proceeds as follows. First a (symmetric) PAKE session is executed using $\mathcal{H}'(pwd)$ as a password on both sides which gives an intermediate PAKE key k. This key is used to derive two independent keys k' and k'' and the client is given $\mathsf{Enc}_{k'}(\mathsf{Enc}_{pwd}(sk))$. C decrypts sk and sends a signature on the entire protocol transcript. If this signature verifies using pk the server accepts the client. The session key of asymmetric PAKE becomes k''. Ω-method can be applied to obtain asymmetric PwA protocols from symmetric ones, in which case k'' can be omitted.

TAG-BASED (ASYMMETRIC) PwA. In the symmetric case any PwA protocol can be transformed into a tag-based tPwA as follows. Parties on input their tags t first compute $\mathcal{H}_T(pwd, t)$ using a cryptographic hash function $\mathcal{H}_T$, which then serves as a password for the original PwA. Since in Definition 3 the adversary specifies tags upon invocation of an instance any successful CAuth-adversary against tPwA can either be used to break CAuth-security of PwA or to find a collision for $\mathcal{H}$, i.e. $\mathsf{Adv}^{\mathsf{tPwA},\mathcal{A}}_{\mathrm{CAuth}}(\kappa) \leq \mathsf{Adv}^{\mathsf{PwA},\mathcal{A}}_{\mathrm{CAuth}}(\kappa) + q\epsilon_{\mathcal{H}_T}(\kappa)$ in q protocol sessions. A similar trick can be applied to asymmetric PwA that are obtained from symmetric PwA using the aforementioned Ω-method — instead of $\mathcal{H}'(pwd)$ in the initial (symmetric) PwA session parties would use $\mathcal{H}_T(\mathcal{H}'(pwd), t)$. Security of such asymmetric tPwA follows from the security of the symmetric PwA, the Ω-method, and the collision-resistance of $\mathcal{H}_T$.

PkA: (Tag-based) Public Key Authentication Protocol The second sub-protocol is a single-side authentication protocol in the public key setting, denoted PkA. In our generalized MFA model the adversarial advantage against client authentication of PkA becomes $\mathsf{Adv}^{\mathcal{A},\mathsf{PkA}}_{\mathrm{CAuth}}(\kappa) = \mathsf{Adv}^{\mathcal{A},(0,1,0)\text{-}\mathsf{MFA}}_{\mathrm{CAuth}}(\kappa)$.

TAG-BASED PkA. Examples of PkA include classical challenge-response protocols, where S sends a (high-entropy) challenge r to C, and C replies with a function of its secret key, e.g. signature. A generic extension of such PkA protocols with tags, denoted tPkA, uses a cryptographic hash function $\mathcal{H}_T$ and follows immediately from [25] — the challenge r received by C with tag t_{C} is transformed into $r'_{\mathrm{C}} = \mathcal{H}_T(r, t_{\mathrm{C}})$, which is then used to generated response to S where it is verified using $r_{\mathrm{S}} = \mathcal{H}_T(r, t_{\mathrm{S}})$. As shown in [25] this conversion is applicable to various classes of PkA protocols, including those based on zero-knowledge proofs.

BiA: (Tag-based) Biometric-based Authentication Protocol The third sub-protocol is a biometric-based authentication protocol, denoted BiA, in which C authenticates towards S that holds some (possibly blinded) reference template of C. In line with our model (and [32]) we consider that biometric data is public and denote the adversarial advantage against client authentication of BiA as $\mathsf{Adv}_{\mathrm{CAuth}}^{\mathcal{A},\mathsf{BiA}}(\kappa) = \mathsf{Adv}_{\mathrm{CAuth}}^{\mathcal{A},(0,0,1)\text{-}\mathsf{MFA}}(\kappa)$.

Explicit (Tag-based) BiA. An explicit-matching BiA protocol with client authentication can be obtained from our $(0,0,1)$-MFAKE construction in Section 3. Since in BiA we are not concerned with session keys, S holding the client's reference template W_{C} can challenge C with a random high-entropy nonce r. C is then expected to reply with a set of hash values $\{h_i\}_i$ from which strictly less than τ values must differ from the respective $\mathcal{H}(\mathrm{C}, \mathrm{S}, r, W_{\mathrm{C},i}, i)$ values computed by S, where $W_{\mathrm{C},i}$ is the ith bit of W_{C}. This BiA is a challenge-response version of our $(0,0,1)$-MFAKE construction from Section 3 and it can be extended to a tag-based tBiA by requiring that C uses its tag t_{C} as an additional input to $\mathcal{H}$ upon computing every h_i. The tag-based security is then implied by collision-resistance of $\mathcal{H}$.

Remark 3. A generic construction of tag-based tBiA out of an arbitrary BiA remains an open problem. Since tags must be processed together with the templates to ensure client authentication, any such transformation would have to deal with the specifics of the underlying matching process, in particular its similarity metric and the form in which W_{C} is stored at S. Even for concrete metrics this task seems challenging. For example, one may try to construct tBiA out of a BiA with the Hamming distance-based metric and explicitly available W_{C} by first, transforming each ith bit of the template into a hash value $h_i = \mathcal{H}(W_{\mathrm{C},i}, t)$ using tag t, and then running the explicit-matching BiA protocol on a (longer) hash string made out of concatenated h_i. For identical reference bits i and tags (on the server and client side) the Hamming distance of the h_i values computed by both parties will be zero. However, for non-matching bits i or tags the resulting h_i values may still match in some (random) fraction of bits. The first problem is to determine an acceptable threshold for the Hamming distance of the resulting hash strings. The second problem is that even an optimal threshold would introduce false negative and positive rates to the matching process.

UKE: Unauthenticated Key Exchange Observe that our previous building blocks do not offer computation of session keys. In our modular (α, β, γ)-MFAKE protocol we will utilize an *unauthenticated* key exchange, denoted UKE, as another sub-protocol. We assume that UKE satisfies the following standard definition (see e.g. [25]) tailored to the client-server scenario.

Definition 4 (Unauthenticated Key Exchange). *An unauthenticated key exchange protocol, denoted* UKE, *is a two-party protocol executed between a client instance* $[\mathrm{C}, s]$ *and a server instance* $[\mathrm{S}, s']$ *such that at the end both instances accept holding respective session keys* k_{C} *and* k_{S} *or reject. Let* $s = \mathsf{tr}_{\mathrm{C}}$ *and* $s' = \mathsf{tr}_{\mathrm{S}}$ *be respective communication transcripts of the two instances. An* UKE *protocol is correct if their partnering, i.e.* $s = s'$, *implies equality of their session keys, i.e.,* $k_{\mathrm{C}} = k_{\mathrm{S}}$.

KE-security. *Consider the following attack game against some correct* UKE *protocol: A PPT adversary* $\mathcal{A}$ *receives as input the security parameter* κ *and can query the* $\mathsf{Transcript}$ *oracle which is parameterized with a random bit* b *fixed in the beginning of the game. On an* i*th query the* $\mathsf{Transcript}$ *oracle executes a protocol session between two new instances of* C *and* S, *and hands its communication transcript* tr_i *and a key* k_i *to* $\mathcal{A}$, *where* k_i *is real if* $b = 0$ *or randomly chosen (for each new* $\mathsf{Transcript}$ *query) if* $b = 1$. *At some point* $\mathcal{A}$ *outputs bit* b'. *An* UKE *protocol is KE-secure if the following advantage is negligible in* κ *for all* $\mathcal{A}$:

$$\mathsf{Adv}_{\mathrm{KE}}^{\mathsf{UKE},\mathcal{A}} = \left| \Pr[b = b'] - \frac{1}{2} \right| .$$

For instance, the standard Diffie-Hellman key exchange protocol in a cyclic group $(\mathbb{G}, g, q)$, where C and S exchange g^x and g^y, respectively, and derive their session keys via $\mathcal{H}(g^x, g^y, g^{xy})$ offers a straightforward KE-secure UKE scheme in the random oracle model under the CDH assumption.

4.3 Modular (α, β, γ)-MFAKE Protocol

We now describe in detail our modular construction of the generalized (α, β, γ)-MFAKE protocol, which supports arbitrary combinations of authentication factors, both in type and quantity.

In addition to the sub-protocols from the previous section, its construction utilizes four hash functions $\mathcal{H}_T, \mathcal{H}_\mathrm{C}, \mathcal{H}_\mathrm{S}, \mathcal{H}_k : \{0,1\}^* \mapsto \{0,1\}^\kappa$, modeled as random oracles that are used for the purpose of tag derivation, key confirmation, and key derivation.

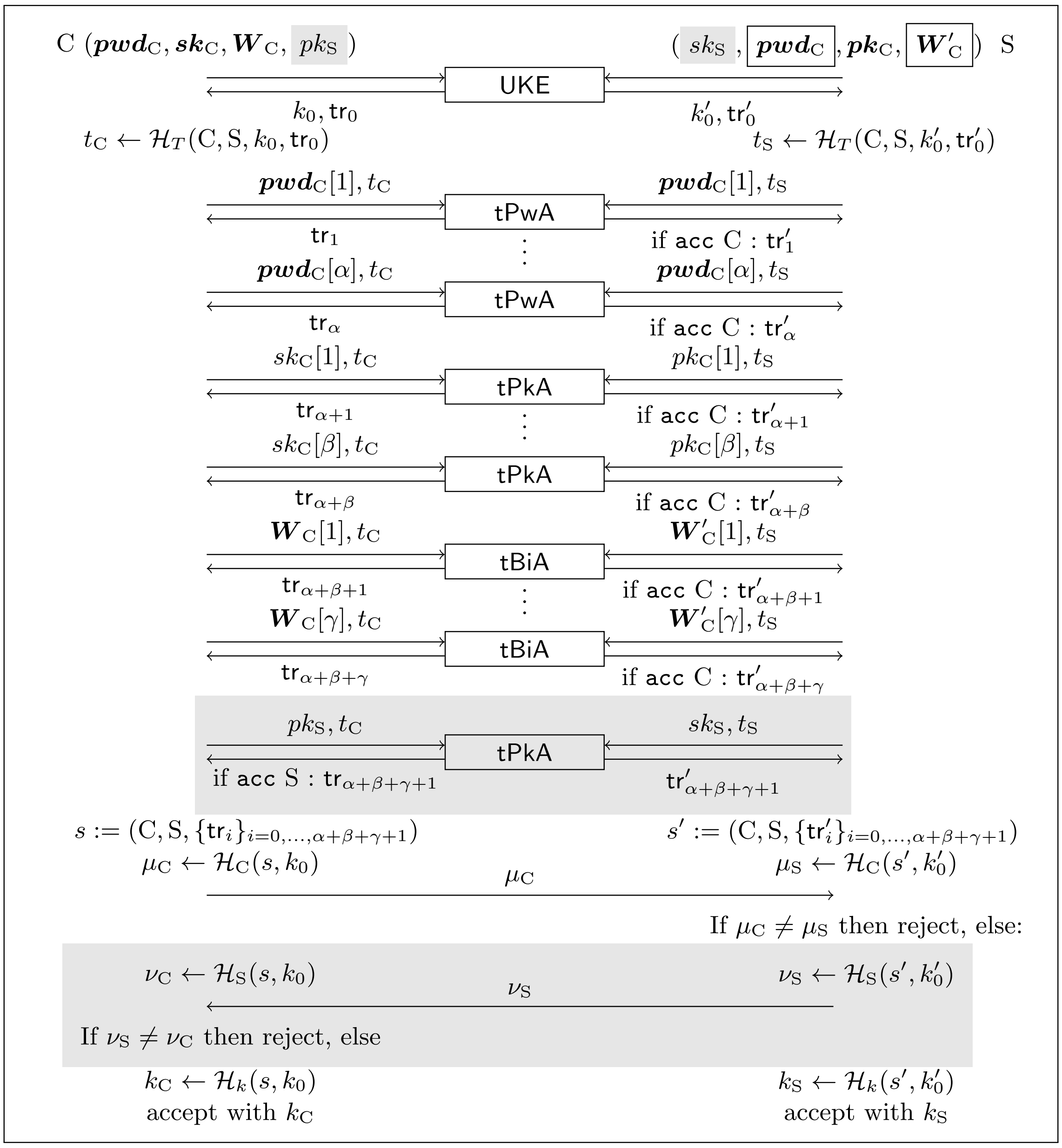
Protocol description. The idea behind our (α, β, γ)-MFAKE design is fairly simple (see illustration in Figure 1): First, C and S run a session of UKE to establish unauthenticated session keys, resulting in k_0 for the client and k_0' for the server, that are then used by both parties to derive their tags (t_C and t_S) through $\mathcal{H}_T$. Then, an appropriate tag-based sub-protocol is executed independently for each authentication factor of the client. That is, C and S execute α sessions of tPwA, β sessions of tPkA (with client-side authentication), and γ sessions of tBiA. S aborts the protocol and rejects C if any of those sessions results in the rejection of the client. The server authentication is optional and is executed through a session of tPkA (with server-side authentication). After finishing all sub-protocols C and S hold their so-far transcripts $\{\mathsf{tr}_i\}_{i=0,\dots\alpha+\beta+\gamma+1}$ and $\{\mathsf{tr}_i'\}_{i=0,\dots\alpha+\beta+\gamma+1}$, respectively, and proceed with the confirmation: C sends a hash value, computed with $\mathcal{H}_\mathrm{C}$, on input its unauthenticated key material from the UKE session as well as session identifier s which comprises transcripts of all executed sub-protocols and the identities of both parties. S verifies that this hash value is as expected. For the optional server authentication, S responds with its own hash value, computed using $\mathcal{H}_\mathrm{S}$ on similar inputs as in the client's case. If the confirmation is successful then parties accept with session keys k_C respectively k_S, derived using $\mathcal{H}_k$.

Instantiations. We can instantiate our generic (α, β, γ)-MFAKE using the examples for its sub-protocols that we gave in Section 4.2. That is, working in prime-order cyclic groups $(\mathbb{G}, g, q)$, we can use unauthenticated Diffie-Hellman key exchange for UKE, a tag-based password-based authentication protocol PwA obtained from the PAKE protocol in [5] (as detailed in Section 4.2), a suitable tag-based challenge-response protocol for tPkA, e.g. using DSS or Schnorr signatures, and our simple tBiA protocol with explicit matching based on the Hamming distance mentioned in Section 4.2. By using the Ω-method from [22] (as also discussed in Section 4.2) we can obtain an asymmetric version of tPwA and use it in our construction. Finally, as evident from the security analysis in Section 4.4, (α, β, γ)-MFAKE can be instantiated from arbitrary sub-protocols as long as those satisfy the required authentication goals. Moreover, as discussed in Section 4.2, tPwA can be obtained generically from PwA, and for a large class of PkA there exists a generic conversion to tPkA. Hence, except for tBiA (cf. Remark 3) all building blocks of (α, β, γ)-MFAKE can be realized using existing efficient (single-factor) authentication solutions.

Generic optimizations. The only dependency amongst the different black-box runs of tag-based authentication sub-protocols is the input tag obtained after UKE session. Therefore, all subsequent sub-protocol runs can be parallelized, resulting in three generic rounds (UKE, tag-based sub-protocols, and confirmation round). Concrete instantiations will often admit further interleaving of messages, e.g. if UKE is instantiated with Diffie-Hellman protocol, S would be able to send all its first messages of tag-based sub-protocols together with its g^y component in the second round. Making a realistic assumption that one-side authenticated tag-based sub-protocols will require two passes each (as in our examples), and interleaving client's μ_C with the pass of the preceding sub-protocol we would end up generically in a three-pass protocol, or a five-pass one if server-side authentication is used. We may reduce costs further by removing some redundancy. For instance, if tPkA and tBiA require random nonces of S then the same nonce could be used for all of them. In a signature-based tPkA we might be able to deploy multi-signatures to reduce the communication costs. In general we expect that our generic construction will perform significantly better than a naïve approach (from introduction) that combines different AKE protocols.

4.4 Security Analysis

We first shed some intuition about the security of the protocol. The actual security is then established by Theorems 1 and 2 that are proven in the appendix.



C ($\boldsymbol{pwd}_{\mathrm{C}}, \boldsymbol{sk}_{\mathrm{C}}, \boldsymbol{W}_{\mathrm{C}}$, pk_{S})
(sk_{S}, $\boxed{\boldsymbol{pwd}_{\mathrm{C}}}$, $\boldsymbol{pk}_{\mathrm{C}}$, $\boxed{\boldsymbol{W}'_{\mathrm{C}}}$) S
UKE
k_0, tr_0
k'_0, tr'_0
$t_{\mathrm{C}} \leftarrow \mathcal{H}_T(\mathrm{C}, \mathrm{S}, k_0, \mathsf{tr}_0)$
$t_{\mathrm{S}} \leftarrow \mathcal{H}_T(\mathrm{C}, \mathrm{S}, k'_0, \mathsf{tr}'_0)$
$\boldsymbol{pwd}_{\mathrm{C}}[1], t_{\mathrm{C}}$
$\boldsymbol{pwd}_{\mathrm{C}}[1], t_{\mathrm{S}}$
tPwA
tr_1
if acc C : tr'_1
$\boldsymbol{pwd}_{\mathrm{C}}[\alpha], t_{\mathrm{C}}$
$\boldsymbol{pwd}_{\mathrm{C}}[\alpha], t_{\mathrm{S}}$
tPwA
tr_α
if acc C : tr'_α
$sk_{\mathrm{C}}[1], t_{\mathrm{C}}$
$pk_{\mathrm{C}}[1], t_{\mathrm{S}}$
tPkA
$\mathsf{tr}_{\alpha+1}$
if acc C : $\mathsf{tr}'_{\alpha+1}$
$sk_{\mathrm{C}}[\beta], t_{\mathrm{C}}$
$pk_{\mathrm{C}}[\beta], t_{\mathrm{S}}$
tPkA
$\mathsf{tr}_{\alpha+\beta}$
if acc C : $\mathsf{tr}'_{\alpha+\beta}$
$\boldsymbol{W}_{\mathrm{C}}[1], t_{\mathrm{C}}$
$\boldsymbol{W}'_{\mathrm{C}}[1], t_{\mathrm{S}}$
tBiA
$\mathsf{tr}_{\alpha+\beta+1}$
if acc C : $\mathsf{tr}'_{\alpha+\beta+1}$
$\boldsymbol{W}_{\mathrm{C}}[\gamma], t_{\mathrm{C}}$
$\boldsymbol{W}'_{\mathrm{C}}[\gamma], t_{\mathrm{S}}$
tBiA
$\mathsf{tr}_{\alpha+\beta+\gamma}$
if acc C : $\mathsf{tr}'_{\alpha+\beta+\gamma}$
$pk_{\mathrm{S}}, t_{\mathrm{C}}$
$sk_{\mathrm{S}}, t_{\mathrm{S}}$
tPkA
if acc S : $\mathsf{tr}_{\alpha+\beta+\gamma+1}$
$\mathsf{tr}'_{\alpha+\beta+\gamma+1}$
$s := (\mathrm{C}, \mathrm{S}, \{\mathsf{tr}_i\}_{i=0,\ldots,\alpha+\beta+\gamma+1})$
$s' := (\mathrm{C}, \mathrm{S}, \{\mathsf{tr}'_i\}_{i=0,\ldots,\alpha+\beta+\gamma+1})$
$\mu_{\mathrm{C}} \leftarrow \mathcal{H}_{\mathrm{C}}(s, k_0)$
$\mu_{\mathrm{S}} \leftarrow \mathcal{H}_{\mathrm{C}}(s', k'_0)$
μ_{C}
If $\mu_{\mathrm{C}} \neq \mu_{\mathrm{S}}$ then reject, else:
$\nu_{\mathrm{C}} \leftarrow \mathcal{H}_{\mathrm{S}}(s, k_0)$
$\nu_{\mathrm{S}} \leftarrow \mathcal{H}_{\mathrm{S}}(s', k'_0)$
ν_{S}
If $\nu_{\mathrm{S}} \neq \nu_{\mathrm{C}}$ then reject, else
$k_{\mathrm{C}} \leftarrow \mathcal{H}_k(s, k_0)$
$k_{\mathrm{S}} \leftarrow \mathcal{H}_k(s', k'_0)$
accept with k_{C}
accept with k_{S}

Fig. 1. Simple (α, β, γ)-MFAKE Protocol. The inputs sk_{S} and pk_{S} are optional for the case of server authentication and so is the server-authenticated execution of tPkA and the confirmation message ν_{S}. These optional parts are shown with a light gray background. Boxed input $\boldsymbol{pwd}_{\mathrm{C}}$ on the server side reflects that client's passwords could be stored in some blinded way, in which case tPwA is assumed to follow the asymmetric setting from [21,22]. Boxed input $\boldsymbol{W}'_{\mathrm{C}}$ on the server side means that client's reference templates are not necessarily stored in clear, in which case tBiA must provide implicit matching functionality.

The initial UKE execution contributes to the forward secrecy of the established session keys. In particular, successful key confirmation guarantees that the corresponding transcripts tr_0 and tr'_0 and the unauthenticated keys k_0 and k'_0 match. Independent runs of tag-based authentication-only protocols for each factor of the client ensure that C was alive at least during that part of the protocol execution. This is because at least one of those factors must remain uncorrupted prior to the acceptance of the server and all sub-protocol transcripts are linked together in the key confirmation step. Since tr_0 and tr'_0 are linked to the transcripts of all authentication-only sub-protocols the key confirmation step further guarantees that the client was alive during the UKE session and, hence, the secrecy of unauthenticated keys k_0 and k'_0 follows from KE-security of the

UKE protocol. The secrecy of k_0 and k_0' carries over to the secrecy of the final session keys k_C and k_S due to the use of independent random oracles. The optional authentication of the server follows the same reasoning as authentication of the client throughout PkA sessions.

Theorem 1. (α,β,γ)-MFAKE *in Figure 1 without server-side authentication is* AKE- *and* CAuth-*secure, in the random oracle model, and*

$$\mathsf{Adv}_{\mathrm{AKE}}^{(\alpha,\beta,\gamma)\text{-}\mathsf{MFAKE},\mathcal{A}}(\kappa) \leq \mathsf{Adv}_{\mathrm{KE}}^{\mathsf{UKE},\mathcal{B}}(\kappa) + \alpha \cdot \mathsf{Adv}_{\mathrm{CAuth}}^{\mathsf{tPwA},\mathcal{B}}(\kappa) + \beta \cdot \mathsf{Succ}_{\mathrm{CAuth}}^{\mathsf{tPkA},\mathcal{B}}(\kappa) + \sum_{i=1}^{\gamma} \mathsf{Adv}_{\mathrm{CAuth}}^{\mathsf{tBiA}_i,\mathcal{B}}(\kappa) + (q_{\mathcal{H}_T}^2 + q(q_{\mathcal{H}_\mathrm{C}} + q_{\mathcal{H}_K})) \cdot 2^{-\kappa}.$$

Theorem 2. (α,β,γ)-MFAKE *in Figure 1 with server-side authentication is* SAuth-*secure, in the random oracle model, and*

$$\mathsf{Succ}_{\mathrm{SAuth}}^{(\alpha,\beta,\gamma)\text{-}\mathsf{MFAKE},\mathcal{A}}(\kappa) \leq \mathsf{Adv}_{\mathrm{KE}}^{\mathsf{UKE},\mathcal{B}}(\kappa) + \mathsf{Succ}_{\mathrm{CAuth}}^{\mathsf{tPkA},\mathcal{B}}(\kappa) + (q_{\mathcal{H}_T}^2 + q(q_{\mathcal{H}_\mathrm{S}} + 1)) \cdot 2^{-\kappa}.$$

5 Conclusion

The proposed generic and modular framework to the design and analysis of multi-factor authentication and key exchange protocols enables black-box constructions of secure MFAKE protocols out of existing and well-understood single-factor authentication-only schemes. The resulting protocol (α,β,γ)-MFAKE enjoys clean design and bears optimization potential since utilized tag-based authentication sub-protocols can be executed in parallel, possibly over different communication channels. The framework sheds light on the relationship between MFAKE and diverse single-factor AKE notions. The modularity of the framework allows for generic accommodation of further authentication factors, e.g. using social relationships [10,15], that may become relevant in the future. An interesting open problem from our work is to come up with a generic extension of biometric-based authentication BiA protocols with tags, in the presence of liveness assumption, which seems challenging as was observed in Remark 3. Although our (α,β,γ)-MFAKE protocol can be instantiated with concrete tag-based tBiA (e.g. the one we discussed), existence of a generic BiA-to-tBiA conversion would allow usage of arbitrary BiA protocols, along with arbitrary PwA and many standard PkA schemes, for which such conversions were demonstrated here resp. in [25].

References

1. *PCI Data Security Standard, Ver.2.* http://www.pcisecuritystandards.org/, 2010.
2. *NIST Special Publication 800-63, Rev.1.* http://csrc.nist.gov/publications/, 2011.
3. M. Abdalla, O. Chevassut, and D. Pointcheval, *One-Time Verifier-Based Encrypted Key Exchange*, in PKC 2005, vol. 3386 of LNCS, Springer, 2005, pp. 47–64.
4. M. Abdalla, P.-A. Fouque, and D. Pointcheval, *Password-Based Authenticated Key Exchange in the Three-Party Setting*, in Public Key Cryptography (PKC 2005), vol. 3386 of LNCS, Springer, 2005, pp. 65–84.
5. M. Abdalla and D. Pointcheval, *Simple Password-Based Encrypted Key Exchange Protocols*, in CT-RSA 2005, vol. 3376 of LNCS, Springer, 2005, pp. 191–208.
6. M. Bellare, D. Pointcheval, and P. Rogaway, *Authenticated Key Exchange Secure against Dictionary Attacks*, in EUROCRYPT 2000, vol. 1807 of LNCS, Springer, 2000, pp. 139–155.
7. M. Bellare and P. Rogaway, *Entity Authentication and Key Distribution*, in CRYPTO 1993, vol. 773 of LNCS, Springer, 1993, pp. 232–249.
8. X. Boyen, *Reusable Cryptographic Fuzzy Extractors*, in ACM CCS 2004, ACM, 2004, pp. 82–91.
9. X. Boyen, Y. Dodis, J. Katz, R. Ostrovsky, and A. Smith, *Secure Remote Authentication Using Biometric Data*, in EUROCRYPT 2005, vol. 3494 of LNCS, Springer, 2005, pp. 147–163.
10. J. G. Brainard, A. Juels, R. L. Rivest, M. Szydlo, and M. Yung, *Fourth-Factor Authentication: Somebody You Know*, in ACM CCS 2006, ACM, 2006, pp. 168–178.
11. J. Bringer and H. Chabanne, *An Authentication Protocol with Encrypted Biometric Data*, in AFRICACRYPT 2008, vol. 5023 of LNCS, Springer, 2008, pp. 109–124.
12. J. Bringer, H. Chabanne, M. Izabachène, D. Pointcheval, Q. Tang, and S. Zimmer, *An Application of the Goldwasser-Micali Cryptosystem to Biometric Authentication*, in ACISP 2007, vol. 4586 of LNCS, Springer, 2007, pp. 96–106.

13. R. Canetti and H. Krawczyk, *Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels*, in EUROCRYPT 2001, vol. 2045 of LNCS, Springer, 2001, pp. 453–474.
14. C. Cremers, *Examining Indistinguishability-Based Security Models for Key Exchange Protocols: The Case of CK, CK-HMQV, and eCK*, in 6th ACM ASIACCS 2011, ACM, 2011, pp. 80–91.
15. E. D. Cristofaro, M. Manulis, and B. Poettering, *Private Discovery of Common Social Contacts*, in ACNS 2011, vol. 6715 of LNCS, 2011, pp. 147–165.
16. Y. Dodis, L. Reyzin, and A. Smith, *Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data*, in EUROCRYPT 2004, vol. 3027 of LNCS, Springer, 2004, pp. 523–540.
17. T. Eisenbarth, T. Kasper, A. Moradi, C. Paar, M. Salmasizadeh, and M. T. M. Shalmani, *On the Power of Power Analysis in the Real World: A Complete Break of the KeeLoqCode Hopping Scheme*, in CRYPTO 2008, vol. 5157 of LNCS, Springer, 2008, pp. 203–220.
18. Federal Financial Institutions Examination Council, *Authentication in an Internet Banking Environment*, 2005. http://www.ffiec.gov/pdf/authentication_guidance.pdf.
19. F. D. Garcia, G. de Koning Gans, R. Muijrers, P. van Rossum, R. Verdult, R. W. Schreur, and B. Jacobs, *Dismantling MIFARE Classic*, in ESORICS 2008, vol. 5283 of LNCS, Springer, 2008, pp. 97–114.
20. F. D. Garcia, P. van Rossum, R. Verdult, and R. W. Schreur, *Dismantling SecureMemory, CryptoMemory and CryptoRF*, in ACM CCS 2010, ACM, 2010, pp. 250–259.
21. C. Gentry, P. D. MacKenzie, and Z. Ramzan, *Password authenticated key exchange using hidden smooth subgroups*, in ACM CCS 2005, ACM, 2005, pp. 299–309.
22. ———, *A Method for Making Password-Based Key Exchange Resilient to Server Compromise*, in CRYPTO 2006, vol. 4117 of LNCS, Springer, 2006, pp. 142–159.
23. F. Hao, *On Robust Key Agreement Based on Public Key Authentication*, in Financial Cryptography and Data Security, vol. 6052 of LNCS, Springer, 2010, pp. 383–390.
24. F. Hao and D. Clarke, *Security Analysis of a Multi-Factor Authenticated Key Exchange Protocol.* Cryptology ePrint Archive, Report 2012/039, 2012. http://eprint.iacr.org/2012/039.
25. T. Jager, F. Kohlar, S. Schäge, and J. Schwenk, *Generic Compilers for Authenticated Key Exchange*, in ASIACRYPT 2010, vol. 6477 of LNCS, Springer, 2010, pp. 232–249.
26. B. A. LaMacchia, K. Lauter, and A. Mityagin, *Stronger Security of Authenticated Key Exchange*, in ProvSec 2007, vol. 4784 of LNCS, 2007, pp. 1–16.
27. Y. Lee, S. Kim, and D. Won, *Enhancement of Two-Factor Authenticated Key Exchange Protocols in Public Wireless LANs*, Computers & Electrical Engineering, 36 (2010), pp. 213–223.
28. C.-T. Li and M.-S. Hwang, *An Efficient Biometrics-Based Remote User Authentication Scheme Using Smart Cards*, Journal of Network and Computer Applications, 33 (2010), pp. 1–5.
29. X. Li, J.-W. Niu, J. Ma, W.-D. Wang, and C.-L. Liu, *Cryptanalysis and Improvement of a Biometrics-Based Remote User Authentication Scheme Using Smart Cards*, Journal of Network and Computer Applications, 34 (2011), pp. 73–79.
30. Y. M. Park and S. K. Park, *Two Factor Authenticated Key Exchange (TAKE) Protocol in Public Wireless LANs*, IEICE Transactions on Communications, E87-B (2004), pp. 1382–1385.
31. K. G. Paterson and D. Stebila, *One-Time-Password-Authenticated Key Exchange*, in ACISP 2010, vol. 6168 of LNCS, Springer, 2010, pp. 264–281.
32. D. Pointcheval and S. Zimmer, *Multi-factor Authenticated Key Exchange*, in ACNS 2008, vol. 5037 of LNCS, Springer, 2008, pp. 277–295.
33. R. Song, *Advanced Smart Card Based Password Authentication Protocol*, Computer Standards & Interfaces, 32 (2010), pp. 321–325.
34. D. Stebila, P. Udupi, and S. Chang, *Multi-Factor Password-Authenticated Key Exchange*, in Eighth Australasian Information Security Conference (AISC 2010), vol. 105, 2010, pp. 56–66.
35. J. E. Tapiador, J. C. Hernandez-Castro, P. Peris-Lopez, and J. A. Clark, *Cryptanalysis of Song's Advanced Smart Card Based Password Authentication Protocol.* arXiv.org, Cryptography and Security, 2011. http://arxiv.org/abs/1111.2744v1.
36. X. Wang and W. Zhang, *An Efficient and Secure Biometric Remote User Authentication Scheme Using Smart Cards*, in Pacific-Asia Workshop on Computational Intelligence and Industrial Application (PACIIA '08), vol. 2, IEEE, 2008, pp. 913–917.

A Proof of Theorem 1 (AKE- and CAuth-security)

Proof of Theorem 1 proceeds in a series of games. These games are written for the AKE-security. To the end of the proof we discuss the impact of game hops on the CAuth-security. We denote by $\mathsf{Succ}^{(\alpha,\beta,\gamma)\text{-}\mathsf{MFAKE},\mathcal{A}}_{\text{AKE-}x}(\kappa)$ the success probability of $\mathcal{A}$ in game G_x. For each game G_x we define $\Delta_x(\kappa)$ as the difference in $\mathcal{A}$'s success probability when playing against the two consecutive games $\mathsf{G}_{\mathsf{x}\text{-}1}$ and G_x, i.e., $\Delta_x(\kappa) = |\mathsf{Succ}^{(\alpha,\beta,\gamma)\text{-}\mathsf{MFAKE},\mathcal{A}}_{\text{AKE-}x}(\kappa) - \mathsf{Succ}^{(\alpha,\beta,\gamma)\text{-}\mathsf{MFAKE},\mathcal{A}}_{\text{AKE-}(x-1)}(\kappa)|$.

G_0 This is the original AKE-security game, where the simulator answers the queries of $\mathcal{A}$ on behalf of the instances according to the specification of (α, β, γ)-MFAKE.

G_1 This game proceeds as G_0, except that for all simulated server and client instances that have matching UKE transcripts $\mathsf{tr}_0 = \mathsf{tr}_0'$ the corresponding UKE keys k_0 and k_0' are chosen at random such that $k_0 = k_0'$ holds. Otherwise, k_0 and k_0' are computed as in G_0.

Claim. $\varDelta_1(\kappa) \le \mathsf{Adv}_{\mathrm{KE}}^{\mathsf{UKE},\mathcal{B}}(\kappa)$. *Proof.* Obviously, for session instances that do not share matching UKE transcripts both games are identical. Any $\mathcal{A}$ that can distinguish between G_1 and G_0 with non-negligible probability can be used to break the KE security from Definition 4. The corresponding KE-adversary $\mathcal{B}$ against UKE would interact with $\mathcal{A}$ and simulate all its (α, β, γ)-MFAKE oracle queries as specified in G_0, except for the messages and keys of the UKE sub-protocol. Assume $\mathcal{A}$ invokes an instance of $\mathrm{U} \in \{\mathrm{C}, \mathrm{S}\}$. If this instance is supposed to send the first message in the UKE session then $\mathcal{B}$ queries its $\mathsf{Transcript}$ oracle and uses the first message of the obtained transcript as a response to $\mathcal{A}$. If $\mathcal{A}$ invokes an instance for $\mathrm{U}' \neq \mathrm{U}$ that is expected to send a message only after having received some incoming message then $\mathcal{B}$ waits for the corresponding Send query of $\mathcal{A}$ and checks whether input message is amongst those output by $\mathcal{B}$ from some transcript that it holds and responds with the next response message from this transcript. If the input message is unexpected then $\mathcal{B}$ runs UKE part on behalf of this instance of U' without consulting its oracle (and will thus be able to compute the UKE key for that session). Once UKE session on behalf of some instance is finished $\mathcal{B}$ has always a key to continue its simulation, either from its own UKE run or from a $\mathsf{Transcript}$ query. The way in which $\mathcal{B}$ simulates UKE sessions ensures that the latter type of keys are used in sessions that involve instances with matching UKE transcripts. If $\mathsf{Transcript}$ returns real keys then we are in G_0; otherwise in G_1. Hence, $\varDelta_1(\kappa) \le \mathsf{Adv}_{\mathrm{KE}}^{\mathsf{UKE},\mathcal{B}}(\kappa)$ as claimed.

G_2 This game proceeds as G_1, except that the simulator aborts if in the ith tPwA session, for some $i \in \{1, \ldots, \alpha\}$, a server instance $[\mathrm{S}, s']$ with tag t_S and (partial) transcript tr_i' accepts client C but there exists no instance of C with matching (partial) transcript tr_i and $t_\mathrm{C} = t_\mathrm{S}$, and $\boldsymbol{pwd}[i]$ is not corrupted.

Claim. $\varDelta_2(\kappa) \le \alpha \cdot \mathsf{Succ}_{\mathrm{CAuth}}^{\mathsf{tPwA},\mathcal{B}}(\kappa)$. *Proof.* We prove this with a hybrid argument, assuming existence of sub-games $\mathsf{G}_2^{\mathsf{upto}(j)}$, $j = 0, \ldots, \alpha$. Let us denote by tPwA_i, $i \in \{1, \ldots, \alpha\}$ the ith tPwA sub-protocol run. In $\mathsf{G}_2^{\mathsf{upto}(j)}$ all tPwA_i, $1 \le i \le j$ are handled as specified in G_2 and all tPwA_i, $j < i \le \alpha$ are handled as specified in G_1. That is, $\mathsf{G}_1 = \mathsf{G}_2^{\mathsf{upto}(0)}$ and $\mathsf{G}_2 = \mathsf{G}_2^{\mathsf{upto}(\alpha)}$. As before, we define $\varDelta_2^{\mathsf{upto}(j)}(\kappa)$ as the difference in $\mathcal{A}$'s success probability in two consecutive games $\mathsf{G}_2^{\mathsf{upto}(j-1)}$ and $\mathsf{G}_2^{\mathsf{upto}(j)}$. The only difference between the two games is that $\mathsf{G}_2^{\mathsf{upto}(j)}$ may still abort even if $\mathsf{G}_2^{\mathsf{upto}(j-1)}$ doesn't. If $\mathcal{A}$ can distinguish between the games then $\mathcal{A}$ must have successfully caused tPwA_j to abort in $\mathsf{G}_2^{\mathsf{upto}(j)}$, in which case an instance $[\mathrm{S}, s']$ accepts C in tPwA_j while no partnered client instance with the same tag exists and no $\mathsf{CorruptClient}(\mathrm{C}, 0, j)$ was asked. Such $\mathcal{A}$ can be immediately used to break CAuth-security of tPwA. The simulator can act as CAuth-adversary $\mathcal{B}$ against tPwA by invoking new instances of the server in the tPwA game using tags of server instances that it simulates in the interaction with $\mathcal{A}$. The simulator relays all tPwA_j related queries of $\mathcal{A}$ as its own queries in the tPwA game and wins if $\mathcal{A}$ causes $\mathsf{G}_2^{\mathsf{upto}(j)}$ to abort. Therefore $\varDelta_2^{\mathsf{upto}(j)}(\kappa) \le \mathsf{Succ}_{\mathrm{CAuth}}^{\mathsf{tPwA},\mathcal{B}}(\kappa)$. By counting hybrids we get $\varDelta_2(\kappa) = \sum_{j=1}^{\alpha} \varDelta_2^{\mathsf{upto}(j)}(\kappa) \le \alpha \cdot \mathsf{Succ}_{\mathrm{CAuth}}^{\mathsf{tPwA},\mathcal{B}}(\kappa)$, as claimed.

G_3 This game proceeds as G_2, except that the simulator aborts if in the ith client side tPkA session, for some $i \in \{1, \ldots, \beta\}$, a server instance $[\mathrm{S}, s']$ with tag t_S and (partial) transcript $\mathsf{tr}_{\alpha+i}'$ accepts client C but there exists no instance of C with matching (partial) transcript $\mathsf{tr}_{\alpha+i}$ and $t_\mathrm{C} = t_\mathrm{S}$, and $\boldsymbol{sk}_\mathrm{C}[i]$ is not corrupted.

Claim. $\varDelta_3(\kappa) \le \beta \cdot \mathsf{Succ}_{\mathrm{CAuth}}^{\mathsf{tPkA},\mathcal{B}}(\kappa)$. *Proof.* We can use essentially the same hybrid argument as in G_2, but for tPkA sessions, and thus build a sequence of β sub-games to show that the difference between any two consecutive sub-games can be upper-bounded by $\mathsf{Succ}_{\mathrm{CAuth}}^{\mathsf{tPkA},\mathcal{B}}(\kappa)$. In this case we obtain $\varDelta_3(\kappa) = \sum_{j=1}^{\beta} \varDelta_3^{\mathsf{upto}(j)}(\kappa) \le \beta \cdot \mathsf{Succ}_{\mathrm{CAuth}}^{\mathsf{tPkA},\mathcal{B}}(\kappa)$, as claimed.

G_4 This game proceeds as G_3, except that the simulator aborts if in the ith tBiA session, for some $i \in \{1, \ldots, \gamma\}$, a server instance $[\mathrm{S}, s']$ with tag t_S and (partial) transcript $\mathsf{tr}'_{\alpha+\beta+i}$ accepts client C but there exists no instance of C with matching (partial) transcript $\mathsf{tr}_{\alpha+\beta+i}$ and $t_\mathrm{C} = t_\mathrm{S}$, and the ith biometric is not corrupted.

Claim. $\Delta_4(\kappa) \leq \sum_{i=1}^{\gamma} \mathsf{Succ}_{\mathrm{CAuth}}^{\mathsf{tBiA}_i,\mathcal{B}}(\kappa)$. *Proof.* We denote by tBiA_i the tBiA protocol operating on the ith biometric. Again, using the hybrid argument as in G_2, but for tBiA sessions, we can build a sequence of γ sub-games and upper-bound the difference between any two consecutive sub-games $\mathsf{G}_4^{\mathsf{upto}(j-1)}$ and $\mathsf{G}_4^{\mathsf{upto}(j)}$ with $\mathsf{Succ}_{\mathrm{CAuth}}^{\mathsf{tBiA}_j,\mathcal{B}}(\kappa)$. The simulator can relay all tBiA_j related queries of $\mathcal{A}$ as its own queries in the tBiA game, including those related to the $\mathsf{BioComp}$ oracle since all biometric-dependent tBiA messages used in the (α, β, γ)-MFAKE protocol remain identical to those of the tBiA protocol. This gives us $\Delta_4(\kappa) = \sum_{j=1}^{\gamma} \Delta_4^{\mathsf{upto}(j)}(\kappa) \leq \sum_{i=1}^{\gamma} \mathsf{Succ}_{\mathrm{CAuth}}^{\mathsf{tBiA}_i,\mathcal{B}}(\kappa)$, as claimed.

Remark 4. Note that if the simulation doesn't abort in this game then it is guaranteed that for each server instance $[\mathrm{S}, s']$ that is entering the confirmation round with partial transcripts $\{\mathsf{tr}'_i\}_{1\leq i\leq \alpha+\beta+\gamma}$ (comprising executions of tPwA, PkA, and tBiA sub-protocols) and tag t_S, and that has not disqualified itself as a candidate for a Test query (i.e. fulfills freshness conditions from Section 2.2), there exists a client instance $[\mathrm{C}, s]$ with partial transcripts $\{\mathsf{tr}_i\}_{1\leq i\leq \alpha+\beta+\gamma}$ such that there exists an index $i, 1 \leq i \leq \alpha + \beta + \gamma$ with $\mathsf{tr}_i = \mathsf{tr}'_i$. Moreover, it is guaranteed that any such client instance holds tag $t_\mathrm{C} = t_\mathrm{S}$.

G_5 This game proceeds as G_4, except that simulation aborts if an instance $[\mathrm{S}, s']$ enters the confirmation round with partial transcripts tr'_0 and $\{\mathsf{tr}'_i\}_{1\leq i\leq \alpha+\beta+\gamma}$ and there exists $[\mathrm{C}, s]$ with partial transcripts tr_0 and $\{\mathsf{tr}_i\}_{1\leq i\leq \alpha+\beta+\gamma}$ such that for some index $i : \mathsf{tr}_i = \mathsf{tr}'_i$ but $\mathsf{tr}_0 \neq \mathsf{tr}'_0$.

Claim. $\Delta_5(\kappa) \leq q^2_{\mathcal{H}_T} 2^{-\kappa}$. *Proof.* G_4 already ensures that if $[\mathrm{S}, s']$ accepts in all authentication sub-protocols then there exists a client instance with $t_\mathrm{C} = t_\mathrm{S}$. The only difference between the two games is that G_5 may still abort even if G_4 doesn't. If $\mathcal{A}$ can distinguish between the games then $\mathcal{A}$ must have successfully caused the simulator to abort in G_5, in which case $[\mathrm{S}, s']$ and $[\mathrm{C}, s]$ hold tags $t_\mathrm{S} = t_\mathrm{C}$ but $\mathsf{tr}_0 \neq \mathsf{tr}'_0$. We can thus output a collision for $\mathcal{H}_T$. Since $\mathcal{H}_T$ is a random oracle we get $\Delta_5(\kappa) \leq q^2_{\mathcal{H}_T} 2^{-\kappa}$, as claimed.

Remark 5. G_5 implies that any instance $[\mathrm{S}, s']$ that was not disqualified as a candidate for a Test query (upon entering the confirmation round) has a corresponding client instance $[\mathrm{C}, s]$ with the same UKE transcript and at least one matching tag-based sub-protocol transcript.

G_6 This game proceeds as G_5, except that on behalf of an instance $[\mathrm{S}, s']$ that is not disqualified as a candidate for a Test query the simulator computes $\mu_\mathrm{S} \leftarrow \mathcal{H}'_\mathrm{C}(s')$ and $k_\mathrm{S} \leftarrow \mathcal{H}'_K(s')$ using two private random oracles $\mathcal{H}'_\mathrm{C}$ and $\mathcal{H}'_k$, and sets $\mu_\mathrm{C} = \mu_\mathrm{S}$ and $k_\mathrm{C} = k_\mathrm{S}$ for the corresponding $[\mathrm{C}, s]$ that has matching UKE transcript and at least one matching tag-based sub-protocol transcript.

Claim. $\Delta_6(\kappa) \leq q(q_{\mathcal{H}_\mathrm{C}} + q_{\mathcal{H}_K}) \cdot 2^{-\kappa}$. Considering that in the previous game, confirmation values and session keys of $[\mathrm{S}, s']$ were derived through random oracles $\mathcal{H}_\mathrm{C}$ and $\mathcal{H}_k$ on input k'_0 (which is random as ensured by G_1) and the transcript s', any $\mathcal{A}$ that can distinguish between the games must ask at some point a query for $\mathcal{H}_\mathrm{C}$ or $\mathcal{H}_k$ containing k'_0 and s' for any of the q invoked sessions as input. Therefore, as claimed $\Delta_6(\kappa) \leq q(q_{\mathcal{H}_\mathrm{C}} + q_{\mathcal{H}_k}) \cdot 2^{-\kappa}$.

G_6 implies that if $[\mathrm{S}, s']$ accepts and is not disqualified as a candidate for a Test query then k_S is uniformly distributed in the domain of session keys. Hence, the probability of $\mathcal{A}$ to win in G_6 no longer depends on the key, i.e. $\mathcal{A}$ can win in G_6 only by guessing bit b (with probability $\frac{1}{2}$).

Summarizing the probability differences across all games we obtain

$$\begin{aligned}
\mathsf{Adv}_{\mathrm{AKE}}^{(\alpha,\beta,\gamma)\text{-}\mathsf{MFAKE},\mathcal{A}}(\kappa) &= \left| \mathsf{Succ}_{\mathrm{AKE}}^{(\alpha,\beta,\gamma)\text{-}\mathsf{MFAKE},\mathcal{A}}(\kappa) - q\left(\frac{\alpha}{|\mathcal{D}_{\mathtt{pwd}}|} + \sum_{i=1}^{\gamma} \mathsf{false}_i^{\mathrm{pos}}\right) - \frac{1}{2} \right| \\
&= \left| \sum_{i=1}^{6} \Delta_i(\kappa) + \frac{1}{2} - q\left(\frac{\alpha}{|\mathcal{D}_{\mathtt{pwd}}|} + \sum_{i=1}^{\gamma} \mathsf{false}_i^{\mathrm{pos}}\right) - \frac{1}{2} \right|.
\end{aligned}$$

Taking into account that

$$\sum_{i=1}^{6} \Delta_i(\kappa) \leq \mathsf{Adv}_{\mathrm{KE}}^{\mathsf{UKE},\mathcal{B}}(\kappa) + \alpha \cdot \mathsf{Succ}_{\mathrm{CAuth}}^{\mathsf{tPwA},\mathcal{B}}(\kappa) + \beta \cdot \mathsf{Succ}_{\mathrm{CAuth}}^{\mathsf{tPkA},\mathcal{B}}(\kappa) + \sum_{i=1}^{\gamma} \mathsf{Succ}_{\mathrm{CAuth}}^{\mathsf{tBiA}_i,\mathcal{B}}(\kappa) + (q_{\mathcal{H}_T}^2 + q(q_{\mathcal{H}_\mathrm{C}} + q_{\mathcal{H}_K})) \cdot 2^{-\kappa},$$

and that

$$\mathsf{Adv}_{\mathrm{CAuth}}^{\mathsf{tPwA},\mathcal{B}}(\kappa) = \left| \mathsf{Succ}_{\mathrm{CAuth}}^{\mathsf{tPwA},\mathcal{B}}(\kappa) - \frac{q}{|\mathcal{D}_{\mathtt{pwd}}|} \right| \quad \text{and} \quad \mathsf{Adv}_{\mathrm{CAuth}}^{\mathsf{tBiA}_i,\mathcal{B}}(\kappa) = \left| \mathsf{Succ}_{\mathrm{CAuth}}^{\mathsf{tBiA}_i,\mathcal{B}}(\kappa) - q \cdot \mathsf{false}_i^{\mathrm{pos}} \right|$$

we obtain

$$\mathsf{Adv}_{\mathrm{AKE}}^{(\alpha,\beta,\gamma)\text{-}\mathsf{MFAKE},\mathcal{A}}(\kappa) \leq \mathsf{Adv}_{\mathrm{KE}}^{\mathsf{UKE},\mathcal{B}}(\kappa) + \alpha \cdot \mathsf{Adv}_{\mathrm{CAuth}}^{\mathsf{tPwA},\mathcal{B}}(\kappa) + \beta \cdot \mathsf{Succ}_{\mathrm{CAuth}}^{\mathsf{tPkA},\mathcal{B}}(\kappa) + \sum_{i=1}^{\gamma} \mathsf{Adv}_{\mathrm{CAuth}}^{\mathsf{tBiA}_i,\mathcal{B}}(\kappa) + (q_{\mathcal{H}_T}^2 + q(q_{\mathcal{H}_\mathrm{C}} + q_{\mathcal{H}_K})) \cdot 2^{-\kappa},$$

which is negligible by the assumptions on the sub-protocols UKE, tPwA, tPkA, and tBiA.

Proof for CAuth-*security.* With regard to client authentication, consider the above game sequence from the perspective of the CAuth-security game and success probability $\mathsf{Succ}_{\mathrm{CAuth}}^{(\alpha,\beta,\gamma)\text{-}\mathsf{MFAKE},\mathcal{A}}(\kappa)$. Freshness conditions regarding server instances encompass the requirements that are relevant for the CAuth-game. Then, Remark 5 implies that in G_5 for each server instance $[\mathrm{S}, s']$ for which $\mathcal{A}$ could still win the game there exists a client instance $[\mathrm{C}, s]$ with the matching UKE transcript and at least one matching tag-based sub-protocol transcript. In G_6, μ_C and μ_S are computed using private oracle, while for CAuth-security modifications of k_C and k_S are irrelevant. The probability difference to G_5 is thus upper-bounded by $q \cdot q_{\mathcal{H}_\mathrm{C}} \cdot 2^{-\kappa}$.Then, $[\mathrm{S}, s']$ must have received $\mu_\mathrm{C} = \mu_\mathrm{S}$ without having a partnered client instance. That is $\mathcal{A}$ must have asked a Send query containing a value that matches a uniformly distributed μ_S. This happens with probability at most $q \cdot 2^{-\kappa}$ for up to q invoked server instances. We thus obtain a negligible CAuth-advantage

$$\mathsf{Adv}_{\mathrm{CAuth}}^{(\alpha,\beta,\gamma)\text{-}\mathsf{MFA},\mathcal{A}}(\kappa) = \mathsf{Adv}_{\mathrm{AKE}}^{(\alpha,\beta,\gamma)\text{-}\mathsf{MFAKE},\mathcal{A}}(\kappa) - q(q_{\mathcal{H}_k} - 1) \cdot 2^{-\kappa}.$$

□

B Proof of Theorem 2 (SAuth-security)

Proof of Theorem 2 resembles in part our proof of Theorem 1. It proceeds in a series of similar games. We denote by $\mathsf{Succ}_{\mathrm{SAuth}\text{-}x}^{(\alpha,\beta,\gamma)\text{-}\mathsf{MFAKE},\mathcal{A}}(\kappa)$ the success probability of $\mathcal{A}$ in game G_x. For each game G_x, we define $\Delta_x(\kappa)$ as the difference in $\mathcal{A}$'s success probability when playing against the two consecutive games $\mathsf{G}_{\mathsf{x}\text{-}1}$ and G_x, i.e., $\Delta_x(\kappa) = |\mathsf{Succ}_{\mathrm{SAuth}\text{-}x}^{(\alpha,\beta,\gamma)\text{-}\mathsf{MFAKE},\mathcal{A}}(\kappa) - \mathsf{Succ}_{\mathrm{SAuth}\text{-}(x-1)}^{(\alpha,\beta,\gamma)\text{-}\mathsf{MFAKE},\mathcal{A}}(\kappa)|$.

G_0 This is the original SAuth-security game, where the simulator answers the queries of $\mathcal{A}$ on behalf of the instances according to the specification of (α, β, γ)-MFAKE.

G_1 This game proceeds as G_0, except that for all simulated server and client instances that have matching UKE transcripts $\mathsf{tr}_0 = \mathsf{tr}'_0$ the corresponding UKE keys k_0 and k'_0 are chosen at random such that $k_0 = k'_0$ holds. Otherwise, k_0 and k'_0 are computed as in G_0.

Claim. $\Delta_1(\kappa) \leq \mathsf{Adv}_{\mathrm{KE}}^{\mathsf{UKE},\mathcal{B}}(\kappa)$. *Proof.* For client and server instances that do not share matching UKE transcripts both games are identical. Any $\mathcal{A}$ that can distinguish between G_1 and G_0 with non-negligible probability can be used to break the KE security from Definition 4. The description of the UKE adversary is exactly the same as in G_1 from the proof of Theorem 1. Hence, $\Delta_1(\kappa) \leq \mathsf{Adv}_{\mathrm{KE}}^{\mathsf{UKE},\mathcal{B}}(\kappa)$, as claimed.

G_2 This game proceeds as G_1, except that the simulator aborts if in the server-side tPkA session a client instance $[\mathrm{C}, s]$ with tag t_C and (partial) transcript $\mathsf{tr}_{\alpha+\beta+\gamma+1}$ accepts server S but there exists no instance of S with matching (partial) transcript $\mathsf{tr}'_{\alpha+\beta+\gamma+1}$ and tag $t_\mathrm{S} = t_\mathrm{C}$, and sk_S is not corrupted.

Claim. $\Delta_2(\kappa) \le \mathsf{Succ}^{\mathsf{tPkA},\mathcal{B}}_{\mathrm{CAuth}}(\kappa)$. *Proof.* As already described in games G_2 through G_4 in proof of Theorem 1 if $\mathcal{A}$ can distinguish between the two games, it can be immediately used to break CAuth-security of tPkA. (In this game CAuth-security is understood as a security property of PkA in case where the authenticating party is the server S. Recall that PkA offers single-side authentication and was defined from the perspective of an authenticating client. In this game the authenticating party is S but the notion of CAuth-security remains as defined.) Hence, $\Delta_2(\kappa) \le \mathsf{Succ}^{\mathsf{tPkA},\mathcal{B}}_{\mathrm{CAuth}}(\kappa)$, as claimed.

Remark 6. Note that if the simulation doesn't abort in G_2 then it is guaranteed that for each client instance $[\mathrm{C}, s]$ that is entering the confirmation round with partial transcript $\mathsf{tr}_{\alpha+\beta+\gamma+1}$ and tag t_C, there exists a server instance $[\mathrm{S}, s']$ with partial transcript $\mathsf{tr}'_{\alpha+\beta+\gamma+1} = \mathsf{tr}_{\alpha+\beta+\gamma+1}$ and $t_\mathrm{S} = t_\mathrm{C}$ if neither C nor $\mathrm{S} = \mathsf{pid}([\mathrm{C}, s])$ has been fully corrupted.

G_3 This game proceeds as G_2, except that simulation aborts if an instance $[\mathrm{C}, s]$ enters the confirmation round with partial transcripts tr_0 and $\mathsf{tr}_{\alpha+\beta+\gamma+1}$ and there exists $[\mathrm{S}, s']$ with partial transcripts tr'_0 and $\mathsf{tr}'_{\alpha+\beta+\gamma+1}$ such that $\mathsf{tr}_{\alpha+\beta+\gamma+1} = \mathsf{tr}'_{\alpha+\beta+\gamma+1}$ but $\mathsf{tr}_0 \neq \mathsf{tr}'_0$.

Claim. $\Delta_3(\kappa) \le q^2_{\mathcal{H}_T} 2^{-\kappa}$. *Proof.* G_2 already ensures that if $[\mathrm{C}, s']$ accepts in the server side tPkA sub-protocol then there exists a server instance with $t_\mathrm{S} = t_\mathrm{C}$. The only difference between the two games is that G_3 may still abort even if G_2 doesn't. If $\mathcal{A}$ can distinguish between the games then $\mathcal{A}$ must have successfully caused the simulator to abort in G_3, in which case $[\mathrm{C}, s]$ and $[\mathrm{S}, s']$ hold tags $t_\mathrm{C} = t_\mathrm{S}$ but $\mathsf{tr}_0 \neq \mathsf{tr}'_0$. We can thus output a collision for $\mathcal{H}_T$. Since $\mathcal{H}_T$ is a random oracle we get $\Delta_3(\kappa) \le q^2_{\mathcal{H}_T} 2^{-\kappa}$, as claimed.

G_4 This game proceeds as G_3, except that on behalf of an instance $[\mathrm{C}, s]$ for which neither C nor $\mathrm{S} = \mathsf{pid}([\mathrm{C}, s])$ is fully corrupted the simulator computes $\nu_\mathrm{C} \leftarrow \mathcal{H}'_\mathrm{S}(s')$ using a private random oracle $\mathcal{H}'_\mathrm{S}$, and sets $\nu_\mathrm{S} = \nu_\mathrm{C}$ for the corresponding $[\mathrm{S}, s']$ that has matching UKE transcript and matching server-side tPkA sub-protocol transcript.

Claim. $\Delta_4(\kappa) \le q \cdot q_{\mathcal{H}_\mathrm{S}} \cdot 2^{-\kappa}$. *Proof.* Considering that in the previous game, confirmation values of $[\mathrm{C}, s]$ were derived through the random oracle $\mathcal{H}_\mathrm{S}$ on input k_0 (which is random as ensured by G_1) and the transcript s, any $\mathcal{A}$ that can distinguish between the games must ask at some point a query for $\mathcal{H}_\mathrm{S}$ containing k_0 and s for any of the q invoked sessions as input. Therefore, $\Delta_4(\kappa) \le q \cdot q_{\mathcal{H}_\mathrm{S}} \cdot 2^{-\kappa}$, as claimed.

Assume that $\mathcal{A}$ wins in G_4. Then, $[\mathrm{C}, s]$ must have received $\nu_\mathrm{S} = \nu_\mathrm{C}$ without having a partnered server instance. That is, $\mathcal{A}$ must have asked a Send query containing a value that matches a uniformly distributed ν_C. This happens with probability at most $q \cdot 2^{-\kappa}$ for up to q invoked client instances. We thus get

$$\begin{aligned}\mathsf{Succ}^{(\alpha,\beta,\gamma)\text{-}\mathsf{MFAKE},\mathcal{A}}_{\mathrm{SAuth}}(\kappa) &= \sum_{i=1}^{4} \Delta_i(\kappa) + q \cdot 2^{-\kappa} \\ &\le \mathsf{Adv}^{\mathsf{UKE},\mathcal{B}}_{\mathrm{KE}}(\kappa) + \mathsf{Succ}^{\mathsf{tPkA},\mathcal{B}}_{\mathrm{CAuth}}(\kappa) + (q^2_{\mathcal{H}_T} + q(q_{\mathcal{H}_\mathrm{S}} + 1)) \cdot 2^{-\kappa},\end{aligned}$$

which is negligible by the assumptions on the sub-protocols UKE and tPkA. □