Designs, Codes and Cryptography (2024) 92:3085–3124
https://doi.org/10.1007/s10623-024-01429-3

TOPAS **2-pass key exchange with full perfect forward secrecy and optimal communication complexity**

Sven Schäge[1]

Received: 15 September 2022 / Revised: 7 December 2023 / Accepted: 17 May 2024 /
Published online: 27 July 2024

Abstract
We present Transmission optimal protocol with active security (TOPAS), the first key agreement protocol with optimal communication complexity (message size and number of rounds) that provides security against fully active adversaries. The size of the protocol messages and the computational costs to generate them are comparable to the basic Diffie-Hellman protocol over elliptic curves (which is well-known to only provide security against passive adversaries). Session keys are indistinguishable from random keys—even under reflection and key compromise impersonation attacks. What makes TOPAS stand out is that it also features a security proof of full perfect forward secrecy (PFS), where the attacker can *actively* modify messages sent to or from the test-session. The proof of full PFS relies on two new extraction-based security assumptions. It is well-known that existing implicitly-authenticated 2-message protocols like HMQV cannot achieve this strong form of (full) security against active attackers (Krawczyk, Crypto'05). This makes TOPAS the first key agreement protocol with full security against active attackers that works in prime-order groups while having optimal message size. We also present a variant of our protocol, TOPAS+, which, under the Strong Diffie-Hellman assumption, provides better computational efficiency in the key derivation phase. Finally, we present a third protocol termed FACTAS (for factoring-based protocol with active security) which has the same strong security properties as TOPAS and TOPAS+ but whose security is solely based on the factoring assumption in groups of composite order (except for the proof of full PFS).

Keywords Cryptography · Protocol · PFS · TOPAS

Mathematical subject classification 94A60

Communicated by R. Steinfeld.

An extended abstract of this work appeared at CCS 2015 https://dblp.uni-trier.de/rec/conf/ccs/Schage15.html?view=bibtex. This version contains besides TOPAS and TOPAS+, (i) a description of the FACTAS protocol that is based on the factoring assumption along with a security proof and (ii) a formal impossibility result showing that the programmability of the random oracle model is necessary for the proofs provided.

✉ Sven Schäge
s.schage@tue.nl

[1] Department of Mathematics & Computer Science, Eindhoven Technical University, Eindhoven, Netherlands

1 Introduction

Besides encryption systems and digital signatures, key exchange protocols are among the most important building blocks of cryptography. It is well-known that the famous Diffie-Hellman (DH) protocol [15] only provides security against passive attackers. This is why since its introduction in 1976, many works focused on upgrading the DH protocol to also shield it against active attacks while trying to keep the overall efficiency as close as possible to the original protocol. An important step in that direction are authenticated DH-based protocols like MQV [25] and its successor HMQV [24]. As in the basic unauthenticated DH protocol, each message consists of only a single group element and messages can be sent in any order. An important feature of these DH-based protocols is that no long-term secret is required when computing the protocol messages; it is only when the session key is derived that the long-term secrets come into play. This generally makes the computation of protocol messages very efficient. The class of protocols that compute messages in this way (without the use of the long-term secrets) are called "implicitly-authenticated" protocols [24]. Unfortunately, in 2005 Krawczyk presented an attack that shows that *implicitly-authenticated protocols inherently cannot provide forward secrecy against active attackers* [24] (see Appendix E for a summary). Only if the attacker *remains passive* with respect to the test-session, implicitly-authenticated protocols can provide perfect forward secrecy. This passive form of PFS is commonly called weak PFS. We stress that weak forward secrecy is not a satisfying definition of security in practice (see Appendix A for a brief example for such a situation). Ultimately, there is no reason to assume that an otherwise unrestricted adversary (with respect to network control) may just refrain from using its full power. Arguably, weak forward-secrecy has rather been defined to show what protocols like HMQV *can* achieve. This is why Krawczyk proposes an extension of HMQV termed HMQV-C, which comprises *three* message flows (while the second flow now also consists of more than 160 bits) and adds explicit key confirmation to the protocol. This guarantees full-PFS security but decreases the protocol's overall efficiency.

We stress that (full) perfect forward secrecy is an important security property for key exchange protocols and that it is naturally well-supported by the original, unauthenticated Diffie-Hellman protocol. As pointed out in [20], the support of PFS is an important advantage over simple, public-key based session key transport and the main reason for the prevalence of DH-like protocols in protocol suites like SSH, IPsec, and TLS.

The only two-message protocol we are aware of that provides truly satisfactory security guarantees against active attackers while maintaining high efficiency is the modified Okamoto–Tanaka (mOT) protocol by Gennaro, Krawczyk, and Rabin (GKR) [20] (depicted in Fig. 4 of Appendix C). Basically, mOT is an enhanced variant of the classical Okamoto–Tanaka protocol [26] from 1989 that introduces additional hashing operations to protect it against several practical attacks and allows a rigorous proof of security.[1] Like the original Okamoto–Tanaka protocol, mOT is defined in groups of hidden order and its security relies on the RSA assumption. Unfortunately, group elements consist of at least 1024 bits so that the overall number of transmitted bits for the two messages is 2048 bits which is much more than what is possible with the basic DH protocol and protocols like HMQV when defined over prime order elliptic curves. However, the protocol has very good computational efficiency.

It is considered an important open problem to design a protocol with full security (including full PFS) against active attackers *and* optimal communication complexity, i.e. where each

[1] In contrast to the original Okamoto–Tanaka protocol, all identities are hashed before usage and the session key is hashed in the final step.

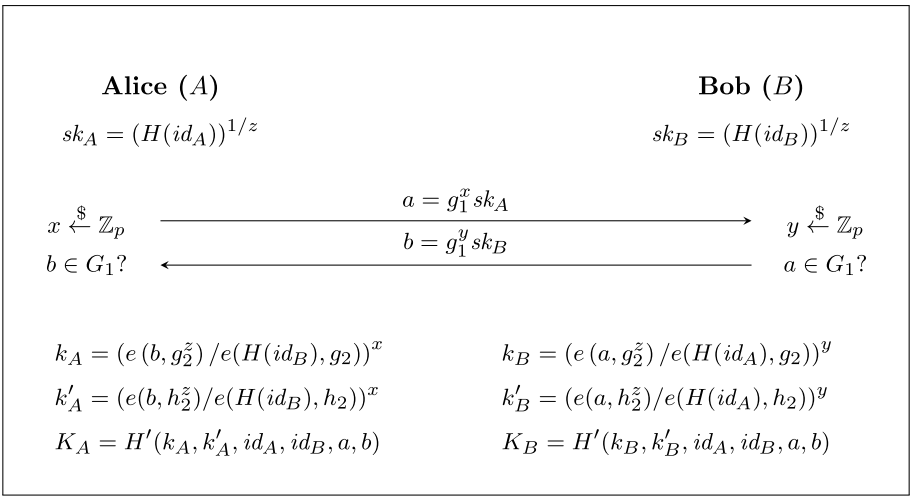

Fig. 1 Overview of TOPAS. The key generation center maintains public parameters *mpk* containing $g_1, g_2, h_2, g_2^z, h_2^z$, prime p, a description of the pairing e, and descriptions of two hash functions $H : \{0,1\}^* \to G_1$ and $H' : \{0,1\}^* \to \{0,1\}^*$. These parameters are available to all parties. We also assume that the identities of all communication partners are publicly available. The master secret *msk* consists of z and is used by the key generation center to derive the user secret keys as $sk_i = (H(id_i))^{1/z}$. K_A (resp. K_B) is the session key computed by Alice (Bob). The pairing operations in the denominator are message-independent and can be pre-computed (in times of low workload) and stored for later use. If Alice also pre-computes $a = g_1^x sk_A$, $(g_2^z)^x$, $(h_2^z)^x$ and $e(H(id_B), g_2)^{-x}$, $e(H(id_B), h_2)^{-x}$ the computation of k, k' will require two pairing operations and two multiplications in G_T per key exchange. Messages can be sent in any order. Without loss of generality we assume that lexically $id_A \leq id_B$. (This is helpful in case the chronological order of the messages is otherwise unclear.)

message only consists of about 160 bits and we have only two messages in total.[2] This is of course optimal, since the birthday bound requires messages to be at least 160 bits for 80 bit security.[3]

Contribution. As our main result, we present TOPAS (short for Transmission Optimal Protocol with Active Security), the first two-message key exchange protocol that provides full perfect forward secrecy and optimal communication complexity (Fig. 1). To achieve this, the design of TOPAS relies on new ideas and techniques. Key indistinguishability, security against key-compromise impersonation (KCI) attacks and reflection attacks are proven under generalizations of the Computational Bilinear Diffie-Hellman Inversion assumption. At the same time, TOPAS is weakly PFS secure under the Computational Bilinear Diffie-Hellman assumption. In Appendix D, we show that all our assumptions are concrete instantiations of the Uber-assumption introduced by Boyen in 2008 and therefore inherit its security in the generic bilinear group model [8]. We stress that for none of our assumptions does the input size grow with the number of adversarial queries (i.e. they do not constitute so-called q-type assumptions). Full-PFS security is shown under two new knowledge-type (or extraction-

[2] Explicit references of the importance of this problem can for example be found http://cyber.biu.ac.il/wp-content/uploads/2017/08/KE2_Hugo_BIU_Feb2018.pdf (p. 31).

[3] Of course, it is necessary that the protocol consists of at least two messages to provide security against active attacks. In any one-message protocol the receiver's computation of the session key can only depend on its knowledge of the secret key (as it cannot feed any session-specific random nonce or ephemeral secrets into the key derivation process). Therefore corrupting the receiver will always reveal the session key (even after the session completes) and PFS is not achievable.

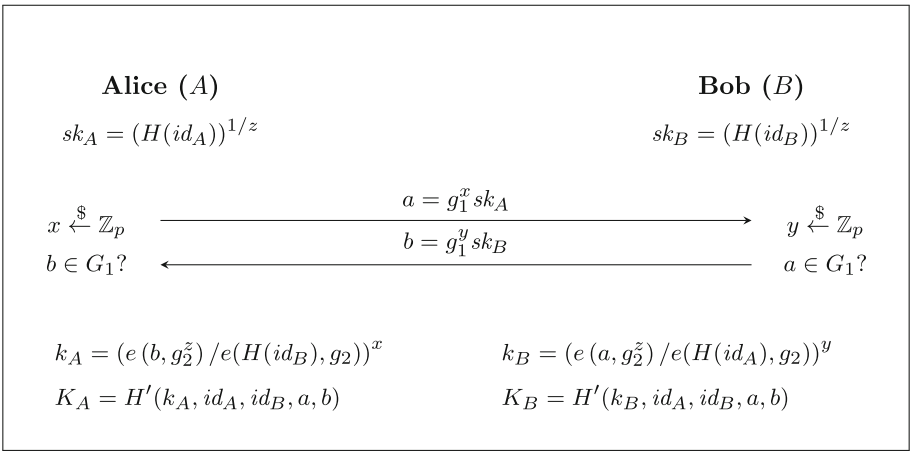

Fig. 2 Overview of TOPAS+. The key generation center maintains public parameters *mpk* containing g_1, g_2, g_2^z, p, a description of the pairing e, and descriptions of two hash functions $H : \{0,1\}^* \to G_1$ and $H' : \{0,1\}^* \to \{0,1\}^*$. These parameters are available to all parties. The master secret *msk* consists of z and is used by the key generation center to derive the user secret keys as $sk_i = (H(id_i))^{1/z}$

type) assumptions that are related to the difficulty of inverting bilinear pairings. (Traditional knowledge-type assumptions are usually related to the difficulty of inverting the modular exponentiation function, i.e. computing discrete logarithms.) Our protocol is defined over asymmetric (Type-3) bilinear groups and all our proofs rely on random oracles. In this work, we assume that the bilinear group supports the ideal ratio of b bit group element size for 2b security. For more conservative choices, the parameters have to be increased correspondingly. When instantiated with our aggressive parameter choices, each message thus consists of only about 160 bits for 80 bit security, resulting in the first key exchange protocol achieving full-PFS with an overall communication complexity of only 320 bits. Moreover, our protocol is identity-based what allows two parties to securely agree on a common session key without a prior exchange of their certificates.

With respect to computational efficiency, we note that all protocol messages can be computed very efficiently, virtually as efficient as in the original DH key exchange. In particular, each message consists of a single ephemeral DH public key that is additionally *multiplied* by the user's secret long-term key. No additional exponentiation is required. Thus the computational overhead when compared to protocols like HMQV is minimal. However, session key derivation in our scheme is comparably slow. This is due to the application of a bilinear pairing in the key derivation process. We note that half of the required pairing operations must only be performed once per communication partner as they only depend on the identity of the communication partner. Finally, we remark that the size of the secret keys derived by the key generation center (KGC) is also only 160 bits using aggressive parameter choices and thus optimal as well.

We also present, TOPAS+ (Fig. 2), a slightly modified version of TOPAS where the security proofs additionally rely on a variant of the Strong Diffie-Hellman assumption [1]. Basically, we require that our generalizations of the Computational Bilinear Diffie-Hellman Inversion assumption remain valid even when the adversary is also given access to an oracle that checks, given input k and $\overline{k}$, whether $k^{z^2} = \overline{k}$ for z unknown to the adversary. The resulting protocol requires less public parameters and only half the number of pairings required to compute

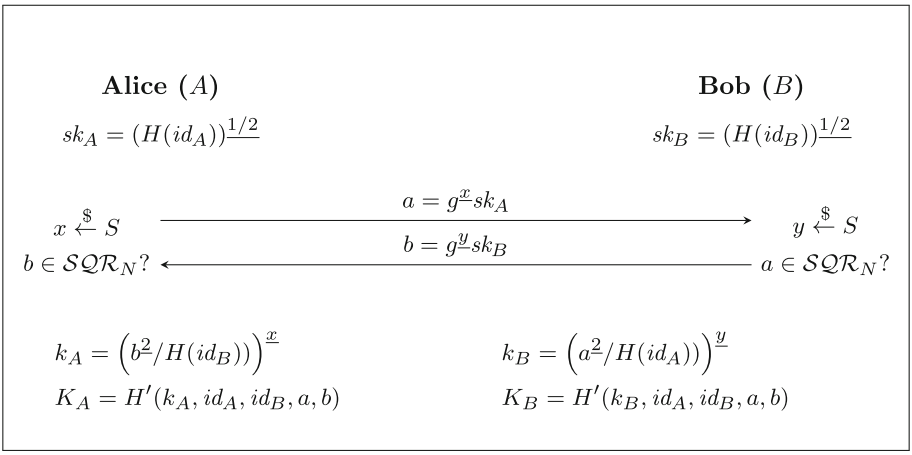

Fig. 3 Overview of FACTAS. The key generation center maintains public parameters *mpk* containing the safe integer $N = p_1 p_2$, $g \in \mathcal{SQR}_N$, and descriptions of two hash functions $H : \{0,1\}^* \to \mathcal{SQR}_N$ and $H' : \{0,1\}^* \to \{0,1\}^*$. These parameters are available to all parties. The master secret *msk* consists of the factorization of N. The set of exponents can be set to $S = [1 \ldots (N-1)/4]$. It can be shown that this distribution is indistinguishable from $S = \left[1 \ldots \left\lfloor \sqrt{N}/2 \right\rfloor\right]$ under the factoring assumption [21]. In [20], the authors recommend a more aggressive choice of S with $S = \left[1 \ldots 2^{2\kappa}\right]$ and additionally assume that this distribution is indistinguishable from the previous ones. The master secret *msk* is used by the key generation center to derive the user secret keys as $sk_i = (H(id_i))^{\underline{1/2}}$, where $g^{\underline{a}} = g^{\underline{a} \bmod |\mathcal{SQR}_N|} = |g^a \underline{\bmod} N|$ is the absolute value of $g^a \underline{\bmod} N \in (-N/2, N/2)$ and elements in $\mathbb{Z}_N^*$ are represented as signed integers in the symmetric interval $(-N/2, N/2)$

the session key. When pre-computing message-independent values offline, key derivation only requires a single pairing operation online. The cost for this modification is that we have to rely on interactive security assumptions even when proving key indistinguishability and security against KCI and reflection attacks.

As our last result, we provide a new protocol that provides full security against active attackers in groups of hidden order (Fig. 3). FACTAS is a variant of the mOT protocol where computations are performed in the group of signed quadratic residues [22]. In contrast to the RSA-based mOT protocol, FACTAS features security reductions to the *factoring* problem in all proofs (except for full PFS). At the same time, the computation of the session key is slightly more efficient than in [20].

We note that due to the application of the bilinear pairing, mOT and FACTAS have a computationally more efficient key derivation than TOPAS and TOPAS+. Concrete numbers can for example be obtained from [28] that compares the properties of a variety of signature schemes on two platforms. (For more details on the exact setup we refer to [28]). The computational costs of the key derivation of mOT are dominated by a variable-base exponentiation with (large) exponents. This is equivalent to the costs incurred by RSA-1028 *signing*. In TOPAS and TOPAS+, the dominant term in the key derivation is the application of the bilinear pairing. The costs are thus comparable to that of BLS signature *verification*. The comparison given in [28] suggests that TOPAS+ protocol has key derivation times roughly 3–4 times slower than mOT.

As mentioned before, the identity-based properties of our protocols avoid that additional information like certificates have to be exchanged between unknown communication part-

ners, in contrast to PKI-based protocols like for example HMQV. This guarantees that in TOPAS and TOPAS+ the size of each message does indeed never exceed 160 bits. Also, the time for key derivation is not slowed down by the additional verification of the received certificate. In general, identity-based protocols greatly pay off in highly dynamic settings where the membership to some eligible group of parties is usually demonstrated via short-lived certificates that are renewed on a regular basis (for example after some weekly or monthly payment).

We also remark that although message computation involves the usage of the secret key, all our protocols provide the strong form of *deniability* defined in [14]. This means that Bob or any other party cannot convince any third party that it once talked to Alice (given that there are no additional side information available to Bob that prove this fact in another way). This is a valuable privacy feature of our protocols that make them suitable for implementing "off-the-record" communication over (insecure) digital networks. We remark that, as with forward secrecy, the basic unauthenticated Diffie-Hellman protocol naturally supports this strong form of deniability (simply because the session key entirely relies only on ephemeral parameters). On the other hand, protocols based on digital signatures (like signed Diffie-Hellman) do not provide such deniability features.

Finally, we note that our proofs of TOPAS and TOPAS+ heavily exploit the programmability of the random oracle model. Using a separation technique that was introduced by Fischlin and Fleischhacker [18] and applied to identity-based non-interactive key exchange by Chen, Huang, and Zhang [11] we can show that, in some sense, the programmability of the random oracle model is actually necessary for our reductions. More concretely, we show in Sect. 6 that under a one-more-type security assumption, the programmability of the random oracle model is necessary for all security proofs that call the adversary once and in a black-box manner, which is the most common type of reduction in cryptography. Unfortunately, the results of [11] cannot directly by applied to TOPAS and TOPAS+ such that we have to rely on new ideas.

Interpretation. From a more theoretical point of view, we continue the work of GKR who analyzed how far the boundaries (in terms of strong security properties) of DH-like protocols can be pushed further while maintaining the efficiency of the original DH protocol (as far as possible). All our protocols add to this body of work as they provide higher efficiency or rely on weaker security assumptions than mOT. In contrast to all other two-message protocols with comparable efficiency that we are aware of—except for the mOT protocol, our protocols are the only ones that provide full security against active attackers. In particular, they show for the first time that protocols in prime-order groups with full security against active adversaries can have optimal message size *and* optimal round complexity and do not necessarily have to rely on costly modifications of the protocol like the addition of key confirmation in HMQV-C.

From a rather orthogonal perspective, our work is strongly motivated by the problem of finding the strongest security properties that two message key exchange protocols *with optimal communication complexity* can achieve. We show for the first time that these protocols can protect against fully active adversaries.

We admit that the feature of full PFS security comes at the cost of relying on (highly) non-standard security assumptions. However, we stress that the existing two-message key exchange protocols with 160 bit messages are implicitly-authenticated and therefore cannot provide full PFS under *any* security assumption.

Application scenarios Our protocols are very interesting for all networks where the *transmission* of data is *very expensive*. Important examples are satellite-based communication networks and communication over low-battery powered wireless (sensor) networks. Satellite-based communication is expensive since all communication flows over a shared

medium with a fixed bandwidth. Using a higher proportion of that bandwidth because of larger cryptographic values will cost more and restrict other useful applications. In wireless sensor networks, node and network lifetimes are highly dependent on energy consumption. This cost is dominated by radio transmission [27, 30, 31]. Besides that, our protocol is of general interest since it is, due to its low message size, less wasteful in terms of network load. Message size has always been an important metric for designers of cryptographic protocols [24] and an important criterion for the selection of modern cryptographic algorithms as new standards. We note that since our protocols are identity-based they ensure that the optimal bound of 160 bits per key exchange message is always met.

SECURITY MODEL To prove security of our protocols we extend and strengthen the security model of mOT [20]. Indistinguishability of session keys from random keys is shown in a variant of the Canetti-Krawczyk (CK) model[9] that is restricted to two message protocols. This variant was first introduced for the analysis of HMQV [24]. The mOT model has further adapted the HMQV model to the identity-based setting. Our model captures security against reflection attacks, key compromise attacks, and forward secrecy. There are two noteworthy ways in which our model differs from [20]. The first is that we provide an explicit Register query to register new users. The second is that we introduce a strengthened definition of weak PFS called *enhanced weak PFS* which allows the adversary to obtain the *secret keys of all parties and the KGC at protocol start-up*, i.e. even before the session key is computed. We note that like the mOT protocol, our protocols require that intermediate values computed in the generation of the protocol messages and the derivation of the session key cannot be revealed by the adversary. Formally, we therefore do not consider state reveal attacks. Technically, this is enforced by requiring that the intermediate values remain in the same protected memory as the long-term key. This is for example similar to DSA signatures, where the random exponent used in the signing procedure must not be revealed to the adversary. Although this seems like a severe restriction, it is, in some sense, the best we can hope for when using two-message protocols. For completeness, in Appendix F we generalize a result by Krawczyk and show that any protocol which allows the adversary to reveal ephemeral keys, cannot provide full PFS. This argument has already been given in [7].

DISCUSSION In the literature, there is a well-known controversy that developed as a reaction to Krawzcyk's arguments[24] showing that implicitly-authenticated protocols inherently cannot provide forward secrecy against active attackers. In particular, there is no consensus on how this result should be interpreted [7]. First, as Cremers and Feltz [16] point out in 2014, the result is only applicable to stateless protocols. Second, beginning with Cremers and Feltz [12], some authors argue for a different way to deal with the inherent technical obstacles that are present in Krawczyk's argument. In a nutshell, instead of giving up on revealing ephemeral keys entirely, [12] propose to use another, more fine-grained notion of full PFS instead. Intuitively, their notion only forbids certain combinations of queries which guarantees to not run into the same technical obstacles. This means that although ephemeral keys are generally allowed, the combinations of queries that are most useful for the attacker (since they provide most of the information) are forbidden. Using their approach, the combination of queries used in Krawczyk's impossibility result is one such particular combination. Accordingly, the model by Cremers and Feltz is theoretically stronger in this sense.

In this paper, we take a more pragmatic approach that, like Krawczyk, largely favors a model that entirely does without ephemeral key reveals. We point out, as Cremers and Feltz have shown, that although revealing some ephemeral keys might not compromise security, which ephemeral keys exactly are unproblematic is highly dependent on the dynamic behavior of the attacker and cannot be planned in advance when setting up parties in real-world systems. From a practical standpoint, it thus seems useful to simply protect all ephemeral keys of the

system sufficiently from being revealed. Now, if this protection mechanism is reliable enough to protect ephemeral keys from being revealed in any problematic combination of highly dynamic attacks, we can as well assume that it rules out ephemeral key reveals entirely. As a benefit, we end up with a simpler security model that unambiguously communicates to developers that additional measures to protect ephemeral keys from being revealed need to be taken.

Related Work It is well-known how to design 2-message protocols that are secure against active adversaries. One way to do this is to add to each (Diffie-Hellman) message a signature that authenticates the originator of that message and protects its integrity [29].[4] This approach has been generalized in [3, 12]. Another solution is to additionally exchange two encrypted nonces that when combined give rise to a symmetric key that is used to protect the integrity of the remaining messages (as used in SKEME [23]). However, all these methods require to send, besides the Diffie-Hellman shares, additional information. For example, consider the most efficient signature scheme that is due to Boneh, Lynn, and Shacham (BLS) where each signature consists of roughly 160 bits. Using the signature-based method with BLS signatures, each party has to exchange considerably more than the optimal amount of bits, namely the key exchange messages plus the size of the signatures (which already account for 160 bits). This does not even consider the costs for certificates that are required when two parties communicate for the first time. At the same time, since these protocols use digital signatures they cannot provide the strong form of deniability given in [14]. Moreover, we remark that when using *any* two-message protocol that provides full PFS we also must have that the corresponding security model does not allow to reveal the ephemeral secrets as formally shown in Appendix F. So, as the protocols in [3, 12] allow the adversary to reveal ephemeral keys, they cannot be shown to provide full PFS in the strong sense of [20].

Another interesting approach is to make practical 2-message protocols like MQV and HMQV identity-based, while keeping their overall efficiency. Most noteworthy, Fiore and Gennaro presented a protocol that features (computational) performance comparable to MQV [17]. However, since it is identity-based there is no need for transmitting certificates as in the original MQV protocol. There are two drawbacks of their protocol. First, each messages consists of two values, thus exceeding the optimum of 160 bits. Second, their protocol does only provide weak PFS, not full PFS. Thus it lacks protection against fully active adversaries. As an advantage, their protocol offers very high computational efficiency.

Identity- based vs. PKI- based Protocols Finally, we would like to comment on the fact that our protocol is identity-based. Our main target is to obtain as short messages as possible while providing high security guarantees. It is interesting to note that when using protocols that provide enhanced weak PFS, the introduction of a KGC does not increase the vulnerability to long-term attacks as compared to relying on classical certification authorities (CAs). As for authentication, any KGC can of course impersonate its users as it can compute their secret keys. However, this is not different from classical CAs that can always create a certificate that binds the identity of the user to a public key chosen by the CA (such that it has access to the corresponding secret key). Now, when it comes to the secrecy of keys of past sessions where the adversary did not actively intervene, our notion of enhanced weak PFS guarantees that even with the help of all user secret keys and even that of the KGC no adversary can obtain the session key. This is exactly what is guaranteed by weak PFS for PKI-based protocols. Indeed we can show that similar to mOT our identity-based protocol can easily be turned into a PKI-based one. Of course we then lose the advantage that users

[4] Obviously, this solution does not preserve the strong deniability property of the original unauthenticated Diffie-Hellman protocol.

need to exchange certificates before communicating for the first time. In Appendix B, we briefly sketch this transformation.

Open problems As stated above, although message computation is quite fast in our first protocol, the computation of the secret key involves costly pairing operations. We leave as an open problem to design a security protocol with optimal communication complexity and more efficient key derivation. We also find it very interesting to design a protocol with similar efficiency while featuring a proof of full PFS that relies on standard assumptions. Finally, we consider it very interesting to design a protocol that provides full PFS while being based on post-quantum security assumptions. Given state-of-the-art techniques, such a protocol is, however, highly likely to exceed the optimal message size that TOPAS and TOPAS+ achieve.

2 Preliminaries

Let κ be the security parameter. Let G_1 and G_2 be groups of prime order p with generators g_1 and g_2 such that $\log_2(p)$ is a polynomial in κ. Let $e : G_1 \times G_2 \rightarrow G_T$ be a non-degenerate bilinear pairing. We call $G = (p, g_1, g_2, e)$ a bilinear group. We will base our protocol on asymmetric bilinear groups of prime order where no isomorphism is known between G_2 and G_1 (Type-3 pairings) [19]. When using asymmetric bilinear groups, we assume that $\log_2(p) \approx 160$ (effectively having $\log_2(p) = 2\kappa$) and that elements of G_1 can be implemented with roughly 160 bits for a security level of approximately 80 bits [2, 4]. In the following we may also refer to what we call an extended bilinear group.

2.1 Security assumptions

In the following, we will present the complexity assumptions that our security analysis of our first protocol relies on. Our main proof will assume the hardness of a generalization of the Computational Bilinear Diffie-Hellman Inversion problem. In Appendix D we will show that all our assumptions are covered by the Uber-assumption introduced in [8] and thus hold in the generic (bilinear) group model. The generic group model is a restricted computational model that idealizes groups G to only allow group operations on the group elements. In such a model it is more easy to derive lower bounds on certain computational tasks since only group operations have to be considered. The underlying assumption is that having access to the concrete representation of group elements in G does not provide additional benefits when attempting to break the computational task. Moreover, a proof in the generic group models is independent of the concrete group instantiation used in practice. The idea is that the proof holds as long as the ultimately used group behaves like a generic group. Groups over elliptic curves are often modeled as generic groups since typically the best-known attacks to solve complexity problems in elliptic curves are generic and could be applied to any other group as well. For the proof of full PFS security we will rely on two new "knowledge-type" (extraction-type) assumptions. We will give a brief motivation for these new assumptions.

(k, l)-Computational Bilinear Diffie- Hellman Inversion ((k, l)-CBDHI) Assumption Let $k = k(\kappa)$ and $l = l(\kappa)$ be polynomials. Assume $G = (p, g_1, g_2, e)$ is a bilinear group. The (k, l)-Computational Bilinear Diffie-Hellman Inversion problem is, given G, the values $g_1^z, g_1^{z^2}, \ldots, g_1^{z^k}$, and $g_2^z, g_2^{z^2}, \ldots, g_2^{z^l}$ for some random $z \in \mathbb{Z}_p$ to compute $e(g_1, g_2)^{1/z}$. This is a generalization of the Computational Bilinear Diffie-Hellman Inversion problem introduced by Boneh–Boyen in [6] where k is fixed to $k = 2$.

Definition 1 (CBDHI Assumption) We say that attacker $\mathcal{A}$ breaks the (k, l)-CBDHI assumption if $\mathcal{A}$ succeeds in solving the (k, l)-Computational Bilinear Diffie-Hellman Inversion problem (where the probability is over the random coins of $\mathcal{A}$ and the random choices for G and z). We say that the (k, l)-CBDHI assumption holds if no PPT attacker $\mathcal{A}$ can break the (k, l)-CBDHI problem.

Looking ahead, in our proof of KCI security we reduce security to the (2, 3)-CBDHI assumption while in our proof of full PFS security we rely on the (3, 3)-CBDHI assumption.

(k, l)-Generalized Computational Bilinear Diffie- Hell - man Inversion ((k, l)-GCBDHI) Assumption Let again $k = k(\kappa)$ and $l = l(\kappa)$ be polynomials in κ and $G = (p, g_1, g_2, e)$ be a bilinear group. The (k, l)-Generalized Computational Bilinear Diffie-Hellman Inversion problem is, given G, random $w \in \mathbb{Z}_p$, $g_1^z, g_1^{z^2}, \ldots, g_1^{z^k}$, and $g_2^z, g_2^{z^2}, \ldots, g_2^{z^l}$ for some random $z \in \mathbb{Z}_p$ to compute $e(g_1, g_2)^{\frac{z+w}{z^2}}$.

Definition 2 (GCBDHI Assumption) We say that attacker $\mathcal{A}$ breaks the (k, l)-GCBDHI assumption if $\mathcal{A}$ succeeds in solving the (k, l)-Generalized Computational Bilinear Diffie-Hellman Inversion problem (where the probability is over the random coins of $\mathcal{A}$ and the random choices for G, z and w). We say that the (k, l)-GCBDHI assumption holds if no PPT attacker $\mathcal{A}$ can break the (k, l)-GCBDHI problem.

We will rely on this assumption for $k = 2$ and $l = 3$ to prove security of our protocol under reflection attacks [24] where the adversary is also allowed to make parties communicate with themselves. We stress that since k, l are constant, the challenge size of both of our assumptions does not grow with the security parameter (and so they do not constitute "q-type" assumptions).

Computational Bilinear Diffie- Hellman (CBDH) Assumption in G_1Assume $G = (p, g_1, g_2, e)$ is a bilinear group. The CBDH problem is, given G and g_1^x, g_1^y to compute $e(g_1, g_2)^{xy}$.

Definition 3 (CBDH Assumption) We say that attacker $\mathcal{A}$ breaks the CBDH assumption if $\mathcal{A}$ succeeds in solving the CBDH problem (where the probability is over the random coins of $\mathcal{A}$ and the random choices for G and x, y). We say that the CBDH assumption holds if no PPT attacker $\mathcal{A}$ can break the CBDH problem.

Later we will use this assumption to prove that our protocol guarantees weak PFS. Observe that the CBDH assumption implies that the classical Computational Diffie-Hellman assumption holds in G_1.

Knowledge of (Pairing) Pre- Image Assumption (KPA) Recall the knowledge of exponent assumption for Diffie-Hellman pairs. It states that for any adversary $\mathcal{A}$ which, given group G (of prime-order p) and two generators $X, Y \in G$ outputs $X', Y' \in G$ such that there is $s \in \mathbb{Z}_p$ with $X' = X^s$ and $Y' = Y^s$, there exists another adversary $\mathcal{A}'$ which given the same inputs additionally outputs the exponent s. However, when working in the target group G_T of a bilinear group this assumption can be false. For example, assume $X = e(A, g_2)$ and $Y = e(B, g_2)$ for some $A, B \in G_1$. Then, an adversary that is given $A, B \in G_1$ and $g_2 \in G_2$ can simply output $X' = e(A, g_2')$ and $Y' = e(B, g_2')$ for some $g_2' \in G_2$ without knowing the discrete logarithm s between X' and Y'.

The following assumption states that although the adversary may not know the discrete logarithm s between X', Y' it must at least know a suitable g_2'. Observe, that if the adversary does indeed know the discrete logarithm s it can easily compute g_2' as $g_2' = g_2^s$. In some sense our new assumption can be viewed as a variant of the knowledge of exponent assumption

(which in its original form is related to the problem of inverting modular exponentiations). However, it is rather a "knowledge of group element" assumption that is related to the difficulty of inverting bilinear pairings.

Formally, security is defined via the following security experiment played between challenger $\mathcal{C}$ and adversary $\mathcal{A}$:

1. $\mathcal{C}$ sends a bilinear group $G = (p, g_1, g_2, e)$ to $\mathcal{A}$ together with $A, B \in G_1$. Let $X = e(A, g_2)$ and $Y = e(B, g_2)$.
2. $\mathcal{A}$ outputs $X', Y' \neq 1_T$.

We say that $\mathcal{A}$ wins if there is some $t \in \mathbb{Z}_p$ with $X' = X^t$ and $Y' = Y^t$.

Definition 4 (Knowledge of Pairing Pre-Image Assumption) We say that the Knowledge of Pairing Pre-Image assumption (KPA) holds, if for every PPT algorithm $\mathcal{A}$ in the above security game there exists another PPT algorithm $\mathcal{A}$' that given the same inputs and random coins as $\mathcal{A}$ behaves exactly like $\mathcal{A}$ (having the same input and output behaviour) while additionally outputting $g_2' = g_2^t$ besides X', Y' such that $X' = e(A, g_2')$ and $Y' = e(B, g_2')$ whenever $\mathcal{A}$ wins.

Modified Knowledge of Co- CDH Assumption The next security assumption we rely on is based on the following problem in bilinear group $G = (p, g_1, g_2, e)$. Assume we provide attacker $\mathcal{A}$ with $A \in G_1$ (such that $A = g_1^a$ for some $a \in \mathbb{Z}_p$) and let $X = e(A, g_2)$. Intuitively, the task of $\mathcal{A}$ is to compute W such that $X = e(A, g_2) = e(g_1, W)$ (i.e. $W = g_2^a$). This is equivalent to solving the Co-CDH assumption [5] in G with challenge A, g_2, g_1. However, in our security experiment we will also give $\mathcal{A}$ access to a Co-CDH oracle. To this end $\mathcal{A}$ may after receiving A specify $Y \in G_T$. As a response $\mathcal{A}$ obtains $U \in G_2$ such that $XY = e(g_1, U)$. The attacker is successful if it can now compute W. We observe that by appropriate choices of Y, $\mathcal{A}$ can easily compute W.

- One way to do this is to have $Y = X^i$ for some $i \neq -1$. We then have that $XY = X^{i+1} = e(g_1, U)$. Therefore, W can simply be computed from U as $W = U^{1/i+1}$.
- Another way is to set $Y = e(g_1, T)$ for some $T \in G_2$ known to $\mathcal{A}$. We then get that $XY = X \cdot e(g_1, T) = e(g_1, U)$ which is equivalent to $X = e(g_1, U/T)$. Thus $W = U/T$ is a correct solution to the problem.

Basically, our new assumption states that every successful adversary must follow one of these strategies—or a combination of both. Intuitively this should still hold if the adversary is, besides U, also provided with $A' = A^{1/z} \in G_2$ (such that $e(A, g_2) = e(A', g_2^z)$) since knowing the z-th root of A for some otherwise unrelated z should not help to break the Co-CDH assumption.

The entire security experiment consists of four steps:

1. $\mathcal{C}$ sends a bilinear group $G = (p, g_1, g_2, e)$ and g_2^z, h_2, h_2^z for uniformly random $z \in \mathbb{Z}_p$ to $\mathcal{A}$ together with uniformly random $A \in G_1$.
2. $\mathcal{A}$ outputs $Y \in G_T$.
3. $\mathcal{C}$ outputs $A^{1/z} \in G_1$ and $U \in G_2$ such that $e(A, g_2) \cdot Y = e(g_1, U)$.
4. $\mathcal{A}$ outputs $W \in G_2$.

$\mathcal{A}$ wins if $e(A, g_2) = e(g_1, W)$.

Definition 5 (MKCoCDH Assumption) We say that the Modified Knowledge of Co-CDH Assumption (MKCoCDH) holds if for every PPT algorithm $\mathcal{A}$ there exists another algorithm $\mathcal{A}$' that given the same inputs and random coins as $\mathcal{A}$ behaves exactly like $\mathcal{A}$ (having the

same input and output behaviour) while in the second step of the above security experiment additionally outputting $i \in \mathbb{Z}_p$ and $T \in G_T$ such that $Y = e(A, g_2)^i \cdot e(g_1, T)$ whenever $\mathcal{A}$ wins.

Now consider a simplified game where less setup parameters are produced by the challenger. (This will later be used in the proof of TOPAS+.)

1. $\mathcal{C}$ sends a bilinear group $G = (p, g_1, g_2, e)$ and g_2^z for uniformly random $z \in \mathbb{Z}_p$ to $\mathcal{A}$ together with uniformly random $A \in G_1$.
2. $\mathcal{A}$ outputs $Y \in G_T$.
3. $\mathcal{C}$ outputs $A^{1/z} \in G_1$ and $U \in G_2$ such that $e(A, g_2) \cdot Y = e(g_1, U)$.
4. $\mathcal{A}$ outputs $W \in G_2$.

$\mathcal{A}$ wins if $e(A, g_2) = e(g_1, W)$.

Definition 6 (MKCoCDH' Assumption) We say that the Simple Modified Knowledge of Co-CDH Assumption (MKCoCDH') holds if for every PPT algorithm $\mathcal{A}$ there exists another algorithm $\mathcal{A}$' that given the same inputs and random coins as $\mathcal{A}$ behaves exactly like $\mathcal{A}$ (having the same input and output behaviour) while in the second step of the above security experiment additionally outputting $i \in \mathbb{Z}_p$ and $T \in G_T$ such that $Y = e(A, g_2)^i \cdot e(g_1, T)$ whenever $\mathcal{A}$ wins.

2.2 Hash functions

Definition 7 (Hash Function) Consider a set $\mathcal{H} = \{H_t\}_{t=1}^{2^\kappa}$ of hash functions indexed by t where each H_t maps from $\{0,1\}^*$ to the hash space T. We require that $\log_2(|T|)$ is a polynomial in κ. We say that $\mathcal{H}$ is collision-resistant if for uniformly random t no PPT attacker can output two distinct string m_1, m_2, such that $H_t(m_1) = H_t(m_2)$ except with negligible probability.

In the following we will always implicitly assume that t is chosen uniformly at random at the beginning of the setup phase. We will then drop t and simply write H (and H'). In the security proofs we model hash functions as random oracles.

2.3 Security model

Let us very briefly re-call the basic features of the security model we use. For a more detailed exposition we refer to [20].

Protocol Framework We consider a set of up to $n = n(\kappa)$ *parties* P_1 to P_n, each of which is identified via unique (identity) strings id_i for $i = 1, \ldots, n$, and a 2-pass key exchange protocol Π that can be run between two parties that we typically denote as id_A and id_B—or Alice and Bob. Unless stated explicitly otherwise we always assume that $id_A \neq id_B$. Each instance of the protocol run at party id_i is called *session* while id_i is also called the *holder* of that session. A session can either *complete*, what involves processing incoming messages and computing outgoing messages until it computes a *session key* K, or it can *abort* in which case no session key will be computed. Additionally we consider *expired* sessions which are completed sessions where the session key and all ephemeral values to compute the session key have been erased.

The party with which the session key is intended to be shared with after the protocol run is called *peer*. (More technically, Bob is the the peer of one of Alice's sessions if that

session uses id_B to derive the session key). The *session identifier* (z_1,z_2,z_3,z_4) of a session is a combination of the identity string of the holder z_1, the identity of the peer z_2, the message sent by the session z_3, and the message received by the session z_4. We say that two sessions *match* if it holds for their session identifiers (z_1,z_2,z_3,z_4) and (z'_1,z'_2,z'_3,z'_4) that $z_1 = z'_2$, $z_2 = z'_1$, $z_3 = z'_4$, and $z_4 = z'_3$. There is also a special party called the *key generation center* that holds a *master secret key msk* and publishes a corresponding *master public key mpk*. The *msk* is used to derive secret keys sk_i for $i = 1, \ldots, n$ for each of the parties from their corresponding identity strings. We assume that each party id_i receives its sk_i from the KGC (in an authentic and confidential way that is out of the scope of this paper). The master public key contains all public information required by the parties to run the protocol. We assume that each party knows all identity strings of the other parties.

ATTACKER We consider an attacker $\mathcal{A}$ that controls the entire network, being able to intercept, modify, drop, replay, and insert messages on transit. To model this, all outgoing messages are delivered to the adversary. If $\mathcal{A}$ only relays all the messages that are sent to some session by its peer it is called *passive* with respect to that session, otherwise it is called *active*. $\mathcal{A}$ can also *activate* sessions of parties to make them engage in a protocol run with peers of $\mathcal{A}$'s choice. To model attack capabilities that grant the adversary access to the secret information of sessions, parties, or the KGC, we allow $\mathcal{A}$ to sent different types of queries to sessions. The following two queries can be used by the attacker to reveal secret values.

- A Reveal query reveals the session key of a complete session.
- A Corrupt query returns all information in the memory of the holder of a session. This includes the secret keys of the party as well as the state information of all its sessions. If a query has been asked to a session with holder id_i we also say that id_i is corrupted.

We say that a session is *exposed* if its holder has been corrupted or its session key been revealed. Additionally sessions are considered exposed if there exists a matching session that is exposed.

Additionally, the following two queries are granted to the attacker.

- A Test query can only be asked once and only to a complete session that is not exposed. Depending on the outcome of a randomly tossed coin $c \in \{0, 1\}$, the output of this query is either the session key K stored at that session (in this context also called the test-session) in case $c = 0$ or a random key uniformly drawn from the space of session keys in case $c = 1$.
- The adversary may also make (up to n) Register queries. On input the j-th identity id_j with $id_j \notin \{id_1, \ldots, id_{j-1}\}$ for $1 \leq j \leq n$, this query creates[5] party P_j and assigns identity id_j to it. Also the secret key sk_j corresponding to identity id_j is given to P_j. We assume that parties are initially uncorrupted.

Observe that in contrast to [20] we have formally introduced a Register query. This models that the adversary may also adaptively choose the identities of the honest (uncorrupted) parties. This is much stronger than in the mOT model, where the identities of the uncorrupted parties are fixed at start-up. (We consider it as an essential feature of identity-based cryptography that the adversary may choose the identities of the honest parties. This is in fact not possible in classical key exchange, where we cannot rule out that when an adversary registers a new public key that it knows the corresponding secret key.) Also, via a combination of Register and Corrupt queries the adversary may obtain secret keys on identities of his choice. The

[5] Alternatively we may think of all the P_i for $1 \leq i \leq n$ to exists before the security game without any identity or secret key. Moreover, they cannot be corrupted. Then Register only assigns id_j and sk_j to P_j.

original model in [24] also specifies queries that reveal the secret state information of sessions. However, as stated before, like mOT, our protocol will not be secure against StateReveal queries (even not when only revealing the ephemeral public keys g_1^x and g_2^y). As in [20] we instead require protection of these values to be at the same level as that of sk_i. As mentioned before we can show in Appendix F that any protocol which allows the adversary to obtain ephemeral secret keys, cannot provide full PFS. In general, we require that except for session keys, all internal information of parties and sessions can only be revealed via full party corruptions.

Security Definitions Let **SG** (short for security game) denote the following security game between a challenger $\mathcal{C}$ and an attacker $\mathcal{A}$.

1. $\mathcal{C}$ gives to $\mathcal{A}$ the master public key mpk.
2. $\mathcal{A}$ may activate sessions and issue Reveal, Corrupt, and Register queries to its liking. Also, $\mathcal{A}$ may use its control of the network to modify messages on transit.
3. $\mathcal{A}$ may ask the Test query to some completed, unexposed session with holder id_A and peer id_B such that $id_A \neq id_B$. Let K be the response and c the internal random coin generated by the test session when answering the query.
4. $\mathcal{A}$ may activate sessions, issue Reveal, Corrupt, and Register queries, and use its control of the network to modify messages on transit.
5. $\mathcal{A}$ outputs $c' \in \{0, 1\}$.

We say that an attacker $\mathcal{A}$ succeeds in a *distinguishing attack* if $c' = c$, the test session is not exposed and the peer of the test-session has not been corrupted.

Definition 8 (Security of Identity-based Key Agreement Protocol) An identity-based key agreement protocol Π is secure if for all PPT attackers $\mathcal{A}$ that are given the above attack capabilities, it holds that i) if two matching sessions of uncorrupted parties complete the probability that the corresponding session keys differ is negligibly close to zero and ii) $\mathcal{A}$ has success probability in a distinguishing attack negligibly close to $1/2$.

Definition 9 (Weak PFS) We say that Π is secure with weak perfect forward secrecy if in **SG** attacker $\mathcal{A}$ is also allowed to corrupt the peer and the holder of the test-session after the test-session key expired and $\mathcal{A}$ has remained passive (only) with respect to the test-session and its matching session(s).

We stress that in our security proof of weak forward secrecy, security even holds when the attacker knows the secret long term keys (but no other session specific secret information) of the peer and the holder and the KGC *before* the session key is computed. This can be interesting when dealing with devices where long-term keys and session specific information are stored separately in two different memories possibly at different locations, but both with approximately the same level of protection against unauthenticated access. Thus corruptions would not reveal session-specific information. In these scenarios the Corrupt query would only allow to reveal the long-term secrets. Next, we present a formal definition that captures this strengthened form of weak PFS. Essentially, it reflects the intuition that forward secrecy should only rely on the secrecy of the ephemeral keys but not of any long-term secret.

Definition 10 (Enhanced Weak PFS) We say that Π provides enhanced weak perfect forward secrecy if Π is secure with weak perfect forward secrecy even if $\mathcal{A}$ is additionally given the secret keys of all parties and the secret key of the KGC at the beginning of the security game and we allow that $id_A = id_B$ for the test-session and its matching session.

Let us now define full PFS. In contrast to the previous definitions we do not require the attacker to remain passive with respect to the test-session.

Definition 11 (Full PFS) We say that Π is secure with full perfect forward secrecy if in **SG** attacker $\mathcal{A}$ is additionally allowed to i) obtain the (long-term) secret key of the holder of the test session at the beginning of the security experiment and ii) corrupt the peer of the test session after the test-session key expired.

Key Compromise Impersonation Attacks We also cover key compromise impersonation attacks[24]. In a KCI attack, $\mathcal{A}$ may after obtaining the secret key of party id_A make id_A falsely believe that it is communicating with some other uncorrupted party id_B although id_A actually isn't. (Obviously impersonating Alice to other parties with the help of Alice's secret key is trivial.)

Definition 12 (KCI Security) We say that Π is secure against KCI attacks if in **SG** attacker $\mathcal{A}$ is additionally allowed to reveal the secret key of the holder of the test session at the beginning of the security experiment (Step 2).

Obviously, KCI security implies security under Definition 8 (Security of Identity-based Key Agreement Protocol) since the adversary is only given additional information to mount its attack.

Reflection Attacks We additionally cover reflection attacks in which an attacker makes two sessions of the same party communicate with each other. As pointed out by [20], these attacks are relevant in real-life scenarios when Alice wants to establish a connection between two of her computers (for example access to a home computer via her laptop).

Definition 13 (Security against Reflection Attacks) We say that Π is secure against reflection attacks if in **SG** attacker $\mathcal{A}$ may also choose a test-session whose peer is equal to its holder, i.e. allowing $id_A = id_B$.

3 Main result

A detailed description of TOPAS is given in Fig. 1. We remark that the challenge in designing a protocol which provides optimal message size and full PFS is that any such protocol must provide two key properties. First, it must include an exchange of ephemeral public keys as otherwise we cannot have any meaningful form of forward secrecy. (Otherwise the session key can be derived by Alice solely from her long-term key and any adversary that obtains this key in a PFS experiment can also compute the session key.) On the other hand, the protocol must also somehow make the parties 'authenticate' their ephemeral public keys using their corresponding long-term secrets as otherwise, by the impossibility result of Krawczyk (Appendix E), we cannot have full PFS. The difficulty when designing a protocol with optimal message length now lies in the fact that we need to combine the two requirements into a single short value.

In TOPAS, Alice and Bob exchange blinded versions of their long-term keys. In particular, in each message, the long-term secret is multiplied by a fresh ephemeral Diffie-Hellman key. Each long-term key in turn is a unique signature on the identity of its holder under the master secret. The verification equation for this signature relies on the bilinear pairing and can be re-written as

$$e(sk_A, g_2^z)/e(H(id_A), g_2) \stackrel{?}{=} 1.$$

The crucial feature of the key derivation of TOPAS is that, due to the bilinearity of the pairing, Bob can remove the signature sk_A (and thus any identity-specific information) from the message $a = g_1^x sk_A$ such that the shared key is independent of sk_A. However, the result lies in the target group and has an additional exponent z:

$$\begin{aligned} e(a, g_2^z)/e(H(id_A), g_2) &= e(g_1^x, g_2^z)e(sk_A, g_2^z)/e(H(id_A), g_2) \\ &= e(g_1^x, g_2^z) = e(g_1, g_2)^{xz}. \end{aligned}$$

By symmetry, Alice computes $e(g_1, g_2)^{yz}$. Together with their own secret ephemeral key, each party can now compute $e(g_1, g_2)^{xyz}$. In the same way, we can obtain

$$\begin{aligned} e(a, h_2^z)/e(H(id_A), h_2) &= e(g_1^x, h_2^z)e(sk_A, h_2^z)/e(H(id_A), h_2) \\ &= e(g_1^x, h_2^z) = e(g_1, h_2)^{xz} \end{aligned}$$

in TOPAS.

In the rest of this section, we present a security analysis of our new protocol. We start by showing that TOPAS provides security against KCI and reflection attacks, as well as enhanced weak PFS under non-interactive security assumptions. Next, we provide a proof of full PFS security.

3.1 Proof of security against KCI and reflection attacks

Before we begin with the formal security proof, let us provide some intuition for the overall strategy. We present a general exposition that is valid for all the protocols presented in this paper and depicted in Figs. 1, 2, and 3. To this end, we introduce some new variables k^* and $\overline{k}$ that will be initialized differently for each of our protocols. As a consequence of this notation, in each protocol, session keys are derived by querying $\hat{k} = (k^*, id_A, id_B, a, b)$ to the random oracle H'. For TOPAS, we define $k^* = (k, k')$ and $\overline{k} = k^{z^2}$. In TOPAS+, we use $k^* = k$ and $\overline{k} = k^{z^2}$ while in FACTAS we will define $k^* = k$ and $\overline{k} = k^4$. We note that k is always part of k^*.

First, as we use H' to compute the final session key, and since H' is modeled as a random oracle, any successful attacker needs to query $\hat{k} = (k^*, id_A, id_B, a, b)$ to the random oracle H' such that $H'(\hat{k}) = K$ (where K is the real session key computed by the test-session) before outputting its guess b'.[6] Informally, the attacker has no information on the session key of a particular session unless it has either asked a Reveal query to that session (or its matching session) or asked $\hat{k}$ to the random oracle H'.

Our overall simulation strategy can be outlined as follows. The simulator chooses setup parameters such that it

1. can compute $\overline{k}$ for any session solely from the session-id (id_A, id_B, a, b), even for the test-session.
2. cannot compute k for the test-session. (Actually, for the proof to go through it is even not necessary that the simulator can compute k for *any* session at all.)
3. can *check* whether for any H' query $\hat{k} = (k^*, id_A, id_B, a, b)$, k^* is indeed the unique intermediate value that occurs in the computation of the session key for session-id (id_A, id_B, a, b). In this context we also say that k^* is correct (with respect to

[6] This is because any adversary not querying the random oracle at $\hat{k}$ has no information on the pre-image of K. This simply follows from the fact that the outputs of a random oracle are chosen at random and independently of the input values—except that they have to be consistent with previous queries. Also, the output space of the hash function is exponential in the security parameter, such that with overwhelming probability no collisions occur after a polynomial number of queries.

(id_A, id_B, a, b)). In particular, the simulator (but not necessarily the attacker) can evaluate an efficient algorithm that given $\overline{k}$ and $\hat{k}$, outputs 1 if k^* is correct with respect to (id_A, id_B, a, b) and 0 otherwise.

4. can extract a solution to the complexity challenge when given the H' query $\hat{k} = (k^*, id_A, id_B, a, b)$ of the test-session, in particular k.

We remark that the condition in 3.1 is very crucial and the main challenge in the security proofs. The simulator must *always* be able to recognize that a given value k (as part of k^*) is exactly the value that a real session would compute. On the other hand, it must not (at least in the test-session) be able to compute the intermediate value k on its own. We need this to recognize if and when the adversary queries a correct k^* to the random oracle. If k^* is correct with respect to the test-session the simulator can use k to extract a solution to the complexity challenge. However it is also important to check if a H' query k^* is correct with respect to some other session as we then have to put additional effort into ensuring that the outputs of the Reveal and H' queries remain consistent. (This is exemplified in Appendix H.3.) Otherwise the adversary could tell the real security game (in which each session always can compute k) from the simulated one (that in contrast sometimes cannot compute k) apart. In essence, we need the underlying problem to behave like a gap-problem, where computing a solution is hard even when given access to a corresponding decision oracle.

In each of our protocols we use a different approach to check whether k^* is correct with respect to some session. In TOPAS, this is ensured via a trapdoor test that was used by Cash, Kiltz, Shoup to reduce the strong twin Diffie-Hellman problem to the ordinary decisional Diffie-Hellman problem [10]. To this end, we require the adversary (and the honest user) to compute, besides k an additional value k' when deriving the session key. Using the trapdoor test and k', the simulator can check whether $k^{z^2} = \overline{k}$. As a result, we can base all security proofs of our first protocol, except for the proof of full PFS, on non-interactive security assumptions. In TOPAS+, we do without the additional value k' at the cost of stronger security assumptions. Here we rely on variants of the Strong Diffie-Hellman assumption. We modify two of the security assumptions that we rely on, the CBDHI assumption and the GCBDHI assumption, to also give the adversary access an oracle $O_{z^2}(\cdot, \cdot)$ that checks whether for a given pair $(\tilde{k}, \tilde{k}^*)$ it holds that $\tilde{k}^* = \tilde{k}^{z^2}$. In this way the simulator can directly use its access to the Strong Diffie-Hellman oracle to check if k^* is correct. As a result, our second protocol has a more efficient key derivation procedure than our first. However, the downside of this approach is that our second protocol relies on interactive security assumptions in all security proofs of security (except for the proof of enhanced weak PFS). In FACTAS, the simulator will simply check whether $(k^*)^{\frac{4}{-}} = \overline{k}$. We stress that for all of our protocols we need that there is no other value $\tilde{k}^* \neq k^*$ which, besides the correct intermediate value k^*, can pass the check. Observe that for all of our protocols this is guaranteed since exponentiation by z^2 constitutes a permutation in prime-order group G_T while exponentiation by 4 represents a permutation in the group of signed quadratic residues $\mathcal{SQR}_N$.

3.2 Basic security properties

Theorem 1 *In the random oracle model,* TOPAS *is secure against KCI attacks under the* (2, 3)-CBDHI *assumption, and secure against reflection attacks under the* (2, 3)-GCBDHI *assumption.*

Proof It is straight-forward to show that two matching sessions compute the same key. Since they are matching, they compute the same session identifier. Also, as shown above they

compute the same values k, k'. Thus all inputs to H' are identical for each session and the session key is equal too.

In the next step, we show that real session keys are indistinguishable from random keys. Assume we are given the random input $G = (p, \hat{g}_1, \hat{g}_2, e)$ and $(\hat{g}_1^t, \hat{g}_1^{t^2}, \hat{g}_2^t, \hat{g}_2^{t^2}, \hat{g}_2^{t^3})$ to the CBDHI/GCBDHI challenge. First, we let the simulator set $g_1 = \hat{g}_1^t$ and $g_2^{z^i} = \hat{g}_2^{t^i}$ for $i = 1, 2, 3$. This implicitly sets $msk = z = t$. Next, the simulator draws random $r, s \in \mathbb{Z}_p$ and sets $h_2 = g_2^s/(g_2^{z^2})^r = g_2^v$ and $h_2^z = (g_2^z)^s/(g_2^{z^3})^r = g_2^{vz}$ for some $v \in \mathbb{Z}_p$. This implicitly sets $v = s - rz^2$. Observe that all values are distributed exactly as in the original security game.

The simulator will randomly choose one party, Bob, to be the peer of the test-session. Since there is only a polynomial number of peers, the simulator's guess is correct with non-negligible probability. Throughout the following, we therefore assume that Bob will not be corrupted by the adversary. Similarly, the simulator will guess the test-session with non-negligible probability.

We will consider two different types of attack strategies. Either the attacker tries to launch a KCI attack or a reflection attack. We exploit that security under KCI attacks implies security in the sense of Definition 8 (Security of Identity-based Key Agreement Protocol). (The only difference is that the attacker may in a KCI attack additionally request the secret key of Alice.) The proofs for both attack types are slightly distinct in the extraction phase. For ease of exposition, we describe a simulation strategy which is for the most part valid for both attack types. We clearly mark when and how the simulation strategies differ in the extraction phase. For better overview, in both cases we always ensure that the peer (Bob) of the test-session which is either held by Alice≠Bob (in case of KCI attacks) or Bob himself (when dealing with reflection attacks) remains uncorrupted. Let us now present the general setup.

SETUP AND SIMULATION OF QUERIES We will first show how the simulator will setup all parameters to be able to answer Corrupt queries for any party except Bob. To this end, the simulator programs the outputs of the random oracle H for all inputs except for id_B as follows: given input id_i it chooses a random value $r_i \in \mathbb{Z}_p$ and outputs $H(id_i) := g_1^{r_i} = \hat{g}_1^{zr_i}$. In this way, the simulator can always compute a corresponding secret key as $sk_i = \hat{g}_1^{r_i}$ and simulate the Register and Corrupt queries. However, for id_B it sets $H(id_B) = \hat{g}_1^{-r_B}$ for some random $r_B \in \mathbb{Z}_p$. Observe that the simulator does not know the corresponding secret key of Bob. In almost all protocol runs the simulator makes sessions (except for those whose holder is Bob) compute their messages and keys as specified in the protocol description. In this way it can also answer all Reveal queries (because the simulator knows the secret key of any party except for Bob).

To compute messages in sessions where Bob is the session holder (we denote the message produced by this session b), the simulator does the following. It chooses a random $b' \in \mathbb{Z}_p$ and computes $b = \hat{g}_1^{b'}$. It then holds that $b^z/H(id_B) = \hat{g}_1^{zb'+r_B} = g_1^{b'+r_B/z}$. Observe that now the secret exponent y in $b = g_1^y (H(id_B))^{1/z}$ is not known to Bob (i.e. the simulator that simulates Bob) as

$$y = b'/z + r_B/z^2$$

and z is unknown. Observe that, as a consequence, the simulator cannot compute k on behalf of Bob anymore when only given message a in case a is produced by the adversary in an active attack.

SIMULATING REVEAL QUERIES FOR BOB Let us show now how the simulator can successfully simulate sessions (and in particular Reveal queries) involving Bob (and the

adversary). To this end we first show that, although the simulator cannot compute k, it can nevertheless always compute $\overline{k} = k^{z^2}$ even when the adversary $\mathcal{A}$ makes Bob engage in a communication with Bob himself. Recall that

$$\overline{k}_B = \left(\frac{e\left(a, g_2^{z^3}\right)}{e\left(H(id_A), g_2^{z^2}\right)} \right)^y .$$

Now, independent of whether a has been computed by Bob (when considering reflection attacks), a session of any other party, or the adversary, the simulator can compute $\overline{k}_B$ for $y = b'/z + r_B/z^2$ as

$$\overline{k}_B = \frac{e\left(a\ ,\ g_2^{yz^3}\right)}{e\left(H(id_A)\ ,\ g_2^{yz^2}\right)} = \frac{e\left(a\ ,\ \left(g_2^{z^2}\right)^{b'} \cdot \left(g_2^{z}\right)^{r_B}\right)}{e\left(H(id_A)\ ,\ \left(g_2^{z}\right)^{b'} \cdot (g_2)^{r_B}\right)} .$$

In the next step, we show that the simulator which knows $\overline{k}$ can check, given k, k', if indeed $\overline{k} = k^{z^2}$ and $k' = k^v$. To this end we apply a variant of the trapdoor test that was introduced in [10]. Recall that we have $h_2 = g_2^v = g_2^s/(g_2^{z^2})^r$ and $h_2^z = g_2^{vz} = (g_2^z)^s/(g_2^{z^3})^r$ for $v = s - rz^2$ unknown to the simulator. We will now show that with overwhelming probability $k^{z^2} = \overline{k} \wedge k^v = k'$ iff $\overline{k}^r k' = k^s$. First assume that $k^{z^2} = \overline{k} \wedge k^v = k'$. Then $\overline{k}^r k' = \left(k^{z^2}\right)^r k^v = \left(k^{z^2}\right)^r k^{s-rz^2} = k^s$ which shows the first direction. Next assume that $\overline{k}^r k' = k^s$. Observe that since $s = v + rz^2$ we get that

$$k^s = k^{v+rz^2} = k^v \left(k^{z^2}\right)^r = k' \overline{k}^r$$

and thus

$$\left(\overline{k}/k^{z^2}\right)^r = k^v/k' \tag{1}$$

while r is information-theoretically hidden from the adversary. Now if $\overline{k} = k^{z^2}$ this must imply $k^v = k'$. In case $\overline{k} \neq k^{z^2}$, $\left(\overline{k}/k^{z^2}\right)^r$ is uniformly distributed in G_T (for random r) while k^v/k' is fixed. Thus the success probability of an adversary to produce k, k' such that Eq. 1 is fulfilled is upper bounded by $1/p$ which is negligible.

So we have now showed that the simulator can always compute $\overline{k}$ and always checks whether a given pair k, k' happens to be "correct" (with respect to some session) in the sense of $k^{z^2} = \overline{k} \wedge k^v = k'$. Let us next describe the strategy of the simulator to program the second random oracle, H', and answer Reveal queries to sessions involving id_B. The main problem is to keep the outputs of the random oracle and the outputs to the Reveal queries consistent. The simulator maintains two lists R and S which are initially both empty. In R we store queries to the random oracle H' and the corresponding answers. In S we simply store session-ids. Let us first describe the basic strategy. Whenever, the attacker queries the random oracle with input x_i we look up if there is some entry (x_i, y_i) already in R. In case it is not, we generate and output a new random string y_i and add (x_i, y_i) to R. If (x_i, y_i) is already in R we output y_i. To compute session-keys for session-id id_A, id_B, a, b we proceed as follows. We look up if there is some entry (u_i, v_i) with $u_i = (id_A, id_B, a, b)$ already in S. In case it is not, we generate and output a new random string v_i and store (u_i, v_i) in S. If (u_i, v_i) is already in S we output v_i. The challenge now is that we have to make sure that the answers stored in S and R remain consistent. In particular, sometimes the outputs stored in S and R must be

identical. (For example, imagine an adversary that successfully computes the values $k, \overline{k}$ of some session with session-id id_A, id_B, a, b. Obviously, querying $x_j = (k, k', id_A, id_B, a, b)$ to the random oracle must produce the same output as when asking the Reveal query to session id_A, id_B, a, b.) To cope with such situations we need to perform additional checks. So whenever we receive a query $x_i = (k, k', id_A, id_B, a, b)$ we additionally check whether there is a corresponding query in S with $u_j = (id_A, id_B, a, b)$ such that $k^{z^2} = \overline{k} \wedge k^v = k'$ for the corresponding $\overline{k}$ value of that session. On success we output $y_i = v_j$ as stored in S. Otherwise we output a random y_i. On the other hand, whenever we encounter a Reveal query for some session held by Bob we can always compute $u_i = (id_A, id_B, a, b)$ and $\overline{k}$. Next we also check whether there is some entry (x_j, y_j) with $x_j = (k, k', id_A, id_B, a, b)$ such that again $k^{z^2} = \overline{k} \wedge k^v = k'$. On success, we output $v_i = y_j$ as stored in R, otherwise we output a random v_i.

EXTRACTION Now that we have showed how to simulate all attack queries, let us proceed to showing how the simulator extracts a solution to the CBDHI challenge. From this point on, we cover KCI attacks and reflection attacks separately. Either the test session is held by Alice≠Bob or Bob.

First we show how the simulator can extract a solution if the test-session is held by Alice. For this session we deviate in the simulation of the test-session from the general simulation strategy that is described above. Instead of generating a honestly as $a = g_1^x H(id_A)^{1/z}$ the simulator computes a as $a = \hat{g}_1^{a'} H(id_A)^{1/z}$ for some random $a' \in \mathbb{Z}_p$. Observe that now the discrete logarithm x in $a = g_1^x H(id_A)^{1/z}$ is implicitly set to $x = a'/z$.

Assume the adversary has non-negligible success probability when querying the Test query to this sessions. In particular, it can decide whether the key provided by the Test query is the real session key or a random key from the same key space. We know that the attacker must ask the correct k, k' values with respect to the test-session to H'. With $y = b'/z + r_B/z^2$ and $x = a'/z$ the simulator in this way obtains k such that

$$\begin{aligned} k &= e(g_1, g_2)^{xyz} = e(\hat{g}_1, \hat{g}_2)^{xyz^2} = e(\hat{g}_1, \hat{g}_2)^{a'/z \cdot (b'/z + r_B/z^2) \cdot z^2} \\ &= e(\hat{g}_1, \hat{g}_2)^{a'b' + a'r_B/z}. \end{aligned}$$

From this we can easily compute a solution d to the CBDHI assumption as

$$d = \left(k e(\hat{g}_1, \hat{g}_2)^{-a'b'} \right)^{1/a'r_B} = e(\hat{g}_1, \hat{g}_2)^{1/z}.$$

Let us now show how to extract a solution to the GCBDHI challenge in case the test-session is held by Bob. Recall that the GCBDHI challenge also contains $w \in \mathbb{Z}_p$ and the task is to compute the value $e(\hat{g}_1, \hat{g}_2)^{\frac{z+w}{z^2}}$. In this case we already have that each message output by Bob is constructed as $b = \hat{g}_1^{b'}$ for random b'. Now for the test-session we slightly deviate and set a as $a = \hat{g}_1^{a'} \hat{g}_1^{r_B}$ for $a' \in \mathbb{Z}_p$ with $a' = r_B w - b'$ (i.e. such that $r_B/(a' + b') = w$). Recall that $H(id_B) = \hat{g}_1^{-r_B}$. This sets x in $a = g_1^x H(id_B)^{1/z} = g_1^x \cdot \hat{g}_1^{-r_B}$ to $x = a' + r_B/z$. Also assume that $b = \hat{g}_1^{b'}$ for random b', implying $y = b'/z + r_B/z^2$. This time the simulator obtains the value k from the queries to the random oracle such that

$$\begin{aligned} k &= e(g_1, g_2)^{xyz} = e(\hat{g}_1, \hat{g}_2)^{xyz^2} \\ &= e(\hat{g}_1, \hat{g}_2)^{(a' + r_B/z) \cdot (b'/z + r_B/z^2) \cdot z^2} \\ &= e(\hat{g}_1, \hat{g}_2)^{a'b' + (a' + b')r_B/z + (r_B)^2/z^2}. \end{aligned}$$

We now easily get a solution to the GCBDHI assumption as

$$\left(ke(\hat{g}_1, \hat{g}_2)^{-a'b'}\right)^{1/(a'+b')r_B} = e(\hat{g}_1, \hat{g}_2)^{\frac{z+r_B/(a'+b')}{z^2}} = e(\hat{g}_1, \hat{g}_2)^{\frac{z+w}{z^2}}.$$

This concludes the proof of security. □

ENHANCED WEAK PFS Let us now show that TOPAS provides enhanced weak PFS. The proof is relatively straight-forward.

Theorem 2 TOPAS *provides enhanced weak perfect forward secrecy under the* CBDH *assumption.*

Proof Except for the generation of two messages a and b, the simulator can setup everything as specified in the protocol description. As before, with non-negligible success probability a is the message sent by the test-session and b is the message received by the test-session. (In contrast to the previous proof the simulator will now also know the secret key of Bob and the master secret z.) Since almost everything is computed as specified in the protocol description and since the session key is expired the simulator can answer all queries of the attacker. We exploit that for enhanced weak PFS security we can assume that a and b may not be produced or modified by the adversary. Let g_1^x, g_1^y be the CBDH challenge. The simulator computes

$$a = g_1^x H(id_A)^{1/z} \text{ and } b = g_1^y H(id_B)^{1/z}.$$

We now have that $k = e(g_1, g_2)^{xyz}$. As before, any successful adversary must query this value to the random oracle H' before answering the test-query. The simulator can guess with non-negligible success probability which of the values queried to H' is equal to k. Then it can simply compute the answer to the CBDH challenge as $k^{1/z} = e(g_1, g_2)^{xy}$. □

3.3 Proof of full PFS security

Theorem 3 TOPAS *provides full PFS under the* (3, 3)-CBDHI*, the* KPA*, and the* MKCoCDH *assumption.*

In contrast to the previous security proof of enhanced weak PFS, the adversary can also modify the messages sent and received by the test-session in the security experiment for full PFS.

Proof Assume there exists an adversary $\mathcal{A}_0$ that breaks the full PFS security of the protocol. In the following we will step-wisely construct a chain of adversaries $\mathcal{A}_0$ to $\mathcal{A}_6$ such that $\mathcal{A}_6$ breaks the (3, 3)-CBDHI assumption. Each adversary $\mathcal{A}_i$ for $i = 1, 2, 3, 4, 5, 6$ is based on the existence of the previous one $\mathcal{A}_{i-1}$.

Let us first recall the essence of the security experiment when proving full PFS security. Besides the setup parameters, the adversary $\mathcal{A}_0$ is also given $a = g_1^x H(id_A)^{1/z}$, sk_A, $H(id_B)$ (but not $(H(id_B))^{1/z}$). In response, the adversary computes $b \in G_1$. Let $Y \in G_1$ be the value such that $b = Y(H(id_B))^{1/z}$. Next, the challenger provides the adversary with $sk_B = (H(id_B))^{1/z}$. Now since the adversary can distinguish K from a random key it must query the corresponding

$$k = \left(e(b, g_2^z)/e(H(id_B), g_2)\right)^x = \left(e(a, g_2^z)/e(H(id_A), g_2)\right)^y$$

to the random oracle H. In the following we always assume, for simplicity, that k is directly given to the challenger.

Attacker $\mathcal{A}_6$ will simulate the real security game to $\mathcal{A}_5$ using a similar setup as in the proofs before. Assume we are given the random CBDHI challenge consisting of $G = (p, \hat{g}_1, \hat{g}_2, e)$ and $(\hat{g}_1^t, \hat{g}_1^{t^2}, \hat{g}_1^{t^3}, \hat{g}_2^t, \hat{g}_2^{t^2}, \hat{g}_2^{t^3})$. Let us first show how the simulator will construct the first part of the public parameters in *mpk* that are to be given to $\mathcal{A}_5$. Again we let the simulator output $g_2^z = \hat{g}_2^t$ as part of *mpk*. Internally, it will also set $g_2^{z^i} = \hat{g}_2^{t^i}$ for $i = 2, 3$. This implicitly sets $msk = z = t$. The simulator draws random $r, s \in \mathbb{Z}_p$ and sets sets $h_2 = g_2^s/(g_2^{z^2})^r = g_2^v$ and $h_2^z = (g_2^z)^s/(g_2^{z^3})^r = g_2^{vz}$ for some $v \in \mathbb{Z}_p$. This implicitly sets $v = s - rz^2$. Observe that all values are distributed exactly as in the original security game.

Next, the simulator draws a random coin $q \in \{0, 1\}$ and a uniformly random $r_B \in \mathbb{Z}_p$. Depending on q, the remaining setup values will slightly differ. That is, the simulator sets

$$H(id_B) = (\hat{g}_1^{t^q})^{r_B} = (\hat{g}_1^{z^q})^{r_B} \text{ and } g_1 = \hat{g}_1^{t^{q+1}} = \hat{g}_1^{z^{q+1}}.$$

Observe that the simulator does not know sk_B in case $q = 0$. However, in case $q = 1$ the simulator knows $sk_B = (H(id_B))^{1/z} = \hat{g}_1^{r_B}$

For $q = 0$ and $q = 1$, the simulator programs the outputs of the random oracle H for all inputs except for id_B as follows: given input id_i (regardless of it being chosen by the adversary as part of a Register query or not) it chooses a random value $r_i \in \mathbb{Z}_p$ and outputs $H(id_i) := g_1^{r_i} = \hat{g}_1^{z^{q+1} r_i}$. In this way, the simulator can always compute a corresponding secret key as $sk_i = \hat{g}_1^{z^q r_i}$ and answer the Corrupt query. As in the previous proofs, all sessions can be simulated with this setup except for the test-session.

Our next goal is to step-wisely construct attacker $\mathcal{A}_5$. It behaves like $\mathcal{A}_0$ but outputs some additional values in case the simulator correctly guesses the test-session (and its peer). We stress that $\mathcal{A}_5$ is an attacker against the full PFS security just like $\mathcal{A}_0$. Let us begin our formal analysis. Assume we have a successful adversary $\mathcal{A}_0$.

ATTACKER $\mathcal{A}_1$ Attacker $\mathcal{A}_1$ will work exactly like $\mathcal{A}_0$ except that it outputs $k^{1/x} = e(b, g_2^z)/e(H(id_B), g_2)$ together with b and g_1, $X = a/sk_A = g_1^x$ at the end of the security game. Observe that these values can easily be computed from the public values alone. In this way, $\mathcal{A}_1$ can easily be computed from $\mathcal{A}_0$.

ATTACKER $\mathcal{A}_2$ Now, since $\mathcal{A}_1$ outputs k, X, $k^{1/x}$, g_1, by the security of the Knowledge of Pairing Pre-Image assumption there also exists an adversary $\mathcal{A}_2$ that works exactly like $\mathcal{A}_1$ except that it also outputs g_2^* together with k such that $k = e(X, g_2^*)$ and $k^{1/x} = e(g_1, g_2^*)$. Since $k = e(g_1, g_2)^{xyz}$, we must have $g_2^* = g_2^{yz}$.

ATTACKER $\mathcal{A}_3$ Next, we show that if $\mathcal{A}_2$ wins the security game against a PFS challenger we can construct an attacker $\mathcal{A}_3$ that can win in the security of the MKCoCDH assumption.

Let us recall the security game of the Modified Knowledge of Co-CDH Assumption. First $\mathcal{A}_3$ receives G, g_2^z, h_2, h_2^z and $B' \in G_1$. Next, $\mathcal{A}_3$ outputs $Y' \in G_T$. As a response, the challenger outputs $B'^{1/z}$ and $U' \in G_2$ with $e(B', g_2) \cdot Y' = e(g_1, U')$. Finally, $\mathcal{A}_3$ outputs W'. It wins if $e(B', g_2) = e(g_1, W')$.

We will now describe how $\mathcal{A}_3$ works by using $\mathcal{A}_2$ as a black-box. First $\mathcal{A}_3$ uses the first part of its input $G = (p, g_1, g_2, e)$, g_2^z, h_2, h_2^z as the public key *mpk* used in the PFS security game when simulating the challenger to $\mathcal{A}_2$. Next, it programs the random oracle H as described before such that all the secret keys $H(id_A)$ for all the identities id_A are known to $\mathcal{A}_3$ except for $id_A = id_B$. Moreover, $\mathcal{A}_3$ sets $H(id_B) := B'$. In this way, $\mathcal{A}_3$ can construct all the values $G, g_2^z, h_2, h_2^z, a = g_1^x H(id_A)^{1/z}, sk_A, H(id_B)$ and simulate the full PFS challenger to $\mathcal{A}_2$.

When $\mathcal{A}_2$ outputs $b, k^{1/x}$. Attacker $\mathcal{A}_3$ sends $Y' = k^{1/x}$ to its challenger. In response $\mathcal{A}_3$ receives $(H(id_B))^{1/z}$ and U' from its challenger. The value $(H(id_B))^{1/z}$ is used as input to $\mathcal{A}_2$. The final output of $\mathcal{A}_2$ is k and $g_2^* = g_2^{yz}$. $\mathcal{A}_3$ can now compute $W' = U'/g_2^{yz}$. Observe that W' is correct since

$$\begin{aligned} e(B', g_2) &= e(g_1, U')/Y' = e(g_1, U')/e(g_1, g_2^*) \\ &= e(g_1, U'/g_2^*) = e(g_1, W'). \end{aligned}$$

So whenever $\mathcal{A}_2$ succeeds in a security game with the PFS challenger so will $\mathcal{A}_3$ in the security game of the Modified Knowledge of Co-CDH Assumption.

ATTACKER $\mathcal{A}_4$ Now by the security of the MKCoCDH assumption, as $\mathcal{A}_3$ succeeds there exists another adversary $\mathcal{A}_4$ that works exactly like $\mathcal{A}_3$ except that it also outputs $i \in \mathbb{Z}_p$, $T \in G_T$ together with Y' such that

$$Y' = e(B', g_2)^i \cdot e(g_1, T).$$

We stress again that in the above series of attackers we have that if $\mathcal{A}_0$ wins so will $\mathcal{A}_4$.

ATTACKER A_5 By the previous arguments, we are guaranteed that attackers $\mathcal{A}_4$ and $\mathcal{A}_2$ exist. Moreover, $\mathcal{A}_4$ can use $\mathcal{A}_2$ as a black-box to win in the MKCoCDH security game whenever $\mathcal{A}_2$ wins.

We will now show an attacker $\mathcal{A}_5$ that uses $\mathcal{A}_4$ and $\mathcal{A}_2$ as a black-box to win against a PFS challenger while outputting additional values besides what is required by definition. In the following, $\mathcal{A}_5$ will play the role of the MKCoCDH challenger against $\mathcal{A}_4$ and play the role of the PFS challenger against $\mathcal{A}_2$. $\mathcal{A}_5$ receives the setup parameters G, g_2^z, h_2, h_2^z, sk_A, $H(id_B)$, and a as input. It relays G, g_2^z, and $B' := H(id_B)$ to $\mathcal{A}_4$. In response, $\mathcal{A}_4$, outputs $k^{1/x}$ together with i, T to $\mathcal{A}_5$. At the same time $\mathcal{A}_5$ sends all values G, g_2^z, h_2^z, h_2^z, sk_A, $H(id_B)$, a to $\mathcal{A}_2$. In response, $\mathcal{A}_2$ outputs b and $k^{1/x}$ to $\mathcal{A}_5$.

The attacker $\mathcal{A}_5$ will output b and i, T, i.e. a mix of the outputs by $\mathcal{A}_4$ and $\mathcal{A}_2$. Next, $\mathcal{A}_5$ receives $(H(id_B))^{1/z}$ from its PFS challenger. Attacker $\mathcal{A}_5$ simply relays this value to $\mathcal{A}_2$. As a response $\mathcal{A}_2$ outputs k and $g_2^* = g_2^{yz}$. These two values are finally output by $\mathcal{A}_5$. Observe that we have not completed the run for $\mathcal{A}_4$. However, we know by our previous analysis that if $\mathcal{A}_2$ is successful, so will $\mathcal{A}_4$ (if we complete the run of $\mathcal{A}_4$). However, at this point it is hidden from $\mathcal{A}_4$'s view that we abort, as all values given to $\mathcal{A}_4$ are distributed exactly as in the real security game. Nevertheless, already at this point we must have that the values i, T are such that $Y' = e(B', g_2)^i \cdot e(g_1, T)$ (otherwise $\mathcal{A}_4$ could not win in case we completed the run with a winning $\mathcal{A}_2$). In all of this, $\mathcal{A}_5$ will deal with any attack queries made by $\mathcal{A}_2$ to its PFS environment by simply relaying them to its own PFS challenger and the corresponding answers back to $\mathcal{A}_2$.

ATTACKER A_6 We will now present an attacker $\mathcal{A}_6$ that can break the CBDHI assumption by using attacker $\mathcal{A}_5$. $\mathcal{A}_6$ will, using the CBDHI challenge, simulate all sessions (except for the test-session) as described before. Let us now turn our attention to the test-session. We have to consider two cases: either it holds for the value i output by $\mathcal{A}_5$ that $i \neq -1$ or $i = -1$.

Let us first consider the case where $i \neq -1$. With probability at least $1/2$ we have that $q = 0$. In this case, it holds that $H(id_B) = \hat{g}_1^{r_B}$. We also have that

$$\frac{e(b, g_2^z)}{e(H(id_B), g_2)} = e(Y, g_2^z) = e(H(id_B), g_2)^i \cdot e(g_1, T).$$

This directly gives

$$\left(\frac{e(b, g_2^z)}{e(g_1, T)}\right)^{1/r_B(i+1)} = e(\hat{g}_1, \hat{g}_2)^{1/z}.$$

It is important to observe that in case $i \neq -1$ the simulator does not need to know $H(id_B)^{1/z}$. It can already break the CBDHI assumption just after receiving b and (i, T).

Now let us turn our attention to the case where $i = -1$. In this case, with probability at least $1/2$ we have that $q = 1$ and $H(id_B) = (\hat{g}^t)^{r_B}$ and $\mathcal{A}_6$ knows $sk_B = \hat{g}^{r_1}$. It can thus successfully send sk_B to the adversary $\mathcal{A}_5$ and receive back g_2^{zy}. Since $i = -1$ we have $e(b, g_2^z) = e(g_1, T)$. It holds that

$$e(b, g_2^z) = e(Yg_1^{r_B/z^2}, g_2^z) = e(g_1, g_2^{zy+r_B/z}) = e(g_1, T).$$

This immediately shows that $T/g_2^{zy} = g_2^{r_B/z}$. We finally get a solution to the complexity challenge as

$$e(\hat{g}_1, T/g_2^{zy})^{1/r_B} = e(\hat{g}_1, \hat{g}_2)^{1/z}.$$

This completes the proof of security. □

4 Higher efficiency

TOPAS+ is a variant of our protocol that features higher efficiency in the key derivation process (Fig. 2). Essentially, it is equivalent to our first protocol except that now only one intermediate value k is computed and fed into the hash function H'. As a consequence we can have a shorter master public key. More importantly, when computing K each party only needs to apply two pairings one of which is message-independent and only needs to be computed once for every communication partner. The security proof of this variant will additionally rely on a variant of the so-called Strong Diffie-Hellman (SDH) assumption. Basically, it states that the assumptions used in the proofs of key indistinguishability, security against reflection, KCI, and full PFS attacks remain valid even if the adversary has access to an oracle $O_{z^2}(\cdot, \cdot)$ with the following property: given $\tilde{k} \in G_T, \tilde{k}^* \in G_T$, $O_{z^2}(\cdot, \cdot)$ outputs 1 iff $\tilde{k}^{z^2} = \tilde{k}^*$ and 0 otherwise.

Definition 14 (CBDHI Assumption) We say that the (k, l)-CBDHI' assumption holds if the (k, l)-CBDHI assumption holds even when the adversary is additionally given access to oracle $O_{z^2}(\cdot, \cdot)$ in the CBDHI security game. Likewise we say that the (k, l)-GCBDHI' assumption holds if the (k, l)-GCBDHI assumption holds even when the adversary is additionally given access to oracle $O_{z^2}(\cdot, \cdot)$ in the GCBDHI security game.

We are now ready to state our result on the security of TOPAS+.

Theorem 4 TOPAS+ *(Fig. 2) has the same security properties under the same security assumptions as* TOPAS *(Fig. 1), except that it relies on the* $(2, 3)$-CBDHI', $(3, 3)$-CBDHI', $(2, 3)$-GCBDHI', *and* MKCoCDH' *assumptions instead of the* $(2, 3)$-CBDHI, $(3, 3)$-CBDHI, $(2, 3)$-GCBDHI, *and* MKCoCDH *assumptions.*

The security proof remains virtually untouched. The only difference is now that we do not need a trapdoor test to maintain consistency when simulating the random oracle. Instead we can directly use the oracle O_{z^2} to check whether a query $k^* = k$ of the adversary (as part of the H' query $\hat{k}$) actually equals the intermediate value computed by some session in the real security game. Again, the simulator is only able to compute $\overline{k} = k^{z^2}$ for all sessions but using $O_{z^2}(\cdot, \cdot)$ it can check if $O_{z^2}(k^*, \overline{k})$ is equal to 1. These modifications affect all proofs except for the proof of enhanced weak PFS.

5 Deniability

Deniable key exchange protocols protects Alice against the unwanted disclosure of her participation in a protocol run via Bob. This can be used to implement a digital variant of "off-the-record" communication over insecure networks. Intuitively, a key exchange protocol provides deniability, if Bob cannot convince a judge, Judy, that Alice once talked to him. To show deniability, it suffices to show that every transcript and corresponding session key that Bob presents to Judy can equally have been produced by a public simulation algorithm that has no access to Alice. More formally, for every PPT Bob that communicates with the PPT Alice, there exists a PPT simulator which when given the same inputs (including the same random coins) as Bob produces transcripts and corresponding session keys which are indistinguishable from those produced by Bob. For a formal treatment of deniability in key exchange protocols see [14]. Implicitly authenticated DH-based protocols like HMQV trivially fulfill this strong form of deniability.

Lemma 1 *Any implicitly authenticated 2-pass protocol meets the strong notion of deniability of [14].*

Proof To see this, observe that Alice's message $a = g^x$ can be computed using public information only (Alice does not use her secret key when computing a). Therefore Bob can compute it in particular. Together with his own message b and his secret key Bob can now easily derive the secret session key. □

In 2-message key exchange protocols where the computation of the exchanged messages involve the secret keys, it may be impossible to achieve deniability. As an example consider exchanging signed DH shares where the signature involves the identities of both parties. Of course, when Bob receives such a signature from Alice and presents it to Judy this immediately proves that Alice once talked to Bob. Fortunately, TOPAS and TOPAS+ provide a very strong form of deniability, although the computation of a involves Alice's secret key.

Theorem 5 TOPAS, TOPAS+, *and* FACTAS *meet the strong notion of deniability of [14].*

Proof Observe that a is uniformly distributed in the underlying group since x is uniform. Therefore the simulator can simulate Alice's message a by just choosing a random group element. Recall that by definition the simulator is also given the same random coins as Bob. Thus and because the simulator also knows Bob's secret key, it can compute y, b and the corresponding session key K in the exact same way as Bob. □

6 On the necessity of the programmable random oracle model

Our proofs of TOPAS and TOPAS+ heavily exploit the programmability of the random oracle model. Using a separation technique that was introduced by Fischlin and Fleischhacker [18] and transferred to the identity-based setting to analyze the Sakai–Ohgishi–Kasahara non-interactive key exchange by Chen, Huang, and Zhang [11] we can show that, in some sense, the programmability of the random oracle model is actually necessary for our reductions.

For preciseness, let us first introduce the notion of simple reductions.

Definition 15 (Simple Reduction) Let R be an efficient security reduction that uses a successful adversary (which is successful in some security game) to break a security assumption.

We say that R is simple if it i) only calls the adversary once, ii) only uses the adversary in a black-box way, and iii) does not rewind the adversary to a point before R has output its first values to the adversary.

Although this restricts the class of possible security reductions, we remark that most reductions in cryptography are simple in the above sense.

Before we can present our impossibility result, we first need to introduce a new security assumption, the CBDHI-2 assumption, that is based on the CBDHI assumption. Consider a discrete logarithm oracle $O_{DL}(\cdot,\cdot)$ which given two group elements G, H in G_T returns the discrete logarithm $x \in \mathbb{Z}_p$ such that $G^x = H$. Basically, our new assumption states that it is hard to break two instances of the CBDHI assumption when given one-time access to $O_{DL}(\cdot,\cdot)$. Similarly, we define CBDHI-2' to be the corresponding assumption for CBDHI'.

Assume $G = (p, g_1, g_2, e)$ is a bilinear group. The (k,l)-CBDHI-2 (respectively (k,l)-CBDHI-2') problem is to solve two random instances (both defined over G) of the (k,l)-CBDHI ((k,l)-CBDHI') problem when additionally given one-time access to $O_{DL}(\cdot,\cdot)$.

Definition 16 (CBDHI-2 and CBDHI'-2 Assumption) We say that attacker $\mathcal{A}$ breaks the (k,l)-CBDHI-2 (respectively the (k,l)-CBDHI-2') assumption if $\mathcal{A}$ succeeds in solving the (k,l)-CBDHI-2 ((k,l)-CBDHI-2') problem (where the probability is over the random coins of $\mathcal{A}$ and the random choices for the bilinear group G and the problem instances z). We say that the (k,l)-CBDHI-2 ((k,l)-CBDHI-2') assumption holds if no PPT attacker $\mathcal{A}$ can break the (k,l)-CBDHI-2 ((k,l)-CBDHI-2') problem.

We remark that these assumptions do not constitute a classical one-more assumption. In particular, the adversary is not provided with an oracle that solves the (k,l)-CBDHI-2 (or (k,l)-CBDHI-2') problem but rather with a full-fledged discrete logarithm oracle. Observe that it is very easy to break a single instance of the CBDHI (CBDHI') problem when given one-time access to $O_{DL}(\cdot,\cdot)$. However, even with one-time access to such an oracle there seems to be no way to solve two instances of the (k,l)-CBDHI-2 ((k,l)-CBDHI-2') problem. We are now able to state our impossibility result.

Theorem 6 *Assume the* $(k,l)-$CBDHI-2 (*respectively* $(k,l)-$CBDHI-$2'$) *assumption holds in the bilinear group G. Then there exists no simple reduction R that reduces the security of* TOPAS (TOPAS+) *to the* (k,l) − CBDHI ((k,l) − CBDHI$'$) *assumption in the non-programmable random oracle model.*

The proof of this theorem is similar to the proof of [11]. The main difference is that we cannot combine two master public keys to form an input for the discrete logarithm oracle such that the output value can be used to efficiently compute secret long-term keys. We can solve this problem by instead combining two long-term secrets that are computed on some random identity. The result is then used as an input to the discrete logarithm oracle.

Proof The idea is to use a so-called meta-reduction that runs the reduction R twice. We denote the specific runs of R as R_1 and R_2. The meta-reduction will simulate theattacker against

R_1 and R_2. The goal of the meta-reduction is to break the $(k, l) - \text{CBDHI-2}$ assumption. Moreover, for the reduction, the simulations must be indistinguishable from real adversaries $\mathcal{A}_1$ and $\mathcal{A}_2$. R_1 is given the first and R_2 the second instance of the CBDHI (CBDHI') problem.

In the following, we assume that in the two runs, reduction R_i for $i \in \{1, 2\}$ computes the master secret z_i. We can now describe two (inefficient) adversaries $\mathcal{A}_1$ and $\mathcal{A}_2$, such that $\mathcal{A}_i$ communicates with the reduction R_i in the security game. Essentially $\mathcal{A}_i$ only accesses the Register and Corrupt queries that are simulated by R_i a few times, modifies message b, and then directly breaks the key indistinguishability of TOPAS (TOPAS+). In particular, after being provided with the master public key, each adversary $\mathcal{A}_i$ calls the Register query and the random oracle on the same five random identities id_0, id_1, id_2, id_3, and id_4. Next, $\mathcal{A}_i$ calls Corrupt on id_0, id_{2i-1}, and id_{2i} and obtains back $sk_0^{(i)}$, $sk_{2i-1}^{(i)}$, and $sk_{2i}^{(i)}$. Using the public parameters $\mathcal{A}_i$ tests whether these secret keys are indeed correct by checking if $e(sk_j^{(i)}, g_2^z) = e(H(id_j), g_2)$, for all $j \in \{0, 2i-1, 2i\}$. On failure $\mathcal{A}_i$ simply aborts. Finally, $\mathcal{A}_i$ computes the secret keys $sk_A^{(i)} := sk_{5-2i}^{(i)}$ and $sk_B^{(i)} := sk_{6-2i}^{(i)}$ of the remaining uncorrupted parties $id_A^{(i)} := id_{5-2i}^{(i)}$ and $id_B^{(i)} := id_{6-2i}^{(i)}$ by brute-force search. After that, $\mathcal{A}_i$ makes the uncorrupted parties id_A and id_B run the protocol. We assume that the oracle of id_A sends the first message a. Next, $\mathcal{A}_i$ intercepts the message b of the oracle of id_B and modifies it to $b' = g^{y_i} sk_B^{(i)}$ for some random but known $y_i \in \mathbb{Z}_p$. In the next step, $\mathcal{A}_i$ calls the Test query on the oracle of id_A to obtain a key candidate. Finally, $\mathcal{A}_i$ breaks the key indistinguishability of the session key by checking if this candidate equals the session key that is derived from a and y_i and the public parameters as given in the key derivation equation. This ends the description of our inefficient adversaries. For contradiction, we now assume that the random oracle H is not programmable, such that the queries to it are always answered in the same way for both adversaries. By definition, the reduction must break the security of the CBDHI (CBDHI') assumption in both runs.

We can then show how we can exploit the reduction to break the CBDHI-2 (CBDHI-2') assumption. To this end, the two instances of the CBDHI-2 (CBDHI-2') assumption are distributed among the two instances of the reduction. Next we show that we can describe a meta-reduction M that manages to efficiently simulate the two adversaries using the $O_{DL}(\cdot, \cdot)$ oracle. M essentially implements two adversaries $\mathcal{A}_1'$ and $\mathcal{A}_2'$ that proceeds exactly as the $\mathcal{A}_i$—except that they do not compute $sk_A^{(i)}$ and $sk_B^{(i)}$ via brute-force search. Instead M calls the discrete logarithm oracle on $H(id_0)$ and computes

$$sk_0^{(2)} / sk_0^{(1)} = H(id_0)^{1/z_2} / H(id_0)^{1/z_1}.$$

As a result, M obtains $c = 1/z_2 - 1/z_1$. With this value, M can now compute the missing secret keys efficiently as

$$\begin{aligned} sk_A^{(i)} &= sk_{5-2i}^{(i)} = sk_{5-2i}^{(3-i)} \cdot H(id_{5-2i})^{(-1)^i c}, \\ sk_B^{(i)} &= sk_{6-2i}^{(i)} = sk_{6-2i}^{(3-i)} \cdot H(id_{6-2i})^{(-1)^i c}. \end{aligned}$$

Essentially M uses the answers to the Corrupt query obtained by $\mathcal{A}_i'$ and the value c to compute the missing secret keys. Observe that the distributions produced by the $\mathcal{A}_i'$ are

identical to those of the $\mathcal{A}_i$. In conclusion, M obtains the two solutions for two instances of the CBDHI-2 (CBDHI-2') instance from R_1 and R_2. At the same time M has only used the discrete logarithm oracle once. This contradicts the security of the CBDHI-2 (CBDHI-2') assumption. Therefore, H cannot be modeled as a non-programmable random oracle. This ends the security proof. □

A On the importance of full PFS in practice

Assume a two-message protocol executed between Alice and Bob where Alice sends a to Bob and Bob sends b to Alice. Now assume that after deriving the secret session key from b and her secret key, Alice immediately sends a sensitive message to Bob that is encrypted as ciphertext c with a key derived from the secret session key. In particular, this message is produced without Alice knowing whether Bob actually has computed the same key. Now assume an adversary who is interested in the contents of the first messages of Alice. Whenever it observes that Alice sends a message a over the network, it drop Bob's b, computes its own value b', and sends it to Alice. Next it intercepts Alice's ciphertext c and records a, b', c in a list. Assume that later on, the adversary learns the secret key of Bob and Alice. Weak perfect forward secrecy does not guarantee the secrecy of the message c because the attacker changed b to b' but full perfect forward secrecy would.

B Sketch of PKI-based protocol variant

TOPAS and TOPAS+ can easily be turned into PKI-based protocols. Due to space limitation we only sketch this here. The global parameters are the public key of the KGC together with g_1^z (this value is not required in our identity-based protocol). The certification authority creates a signature key pair and publishes the public key. User keys are generated as follows. Each user chooses a random $r \in \mathbb{Z}_p$ and computes $sk = g_1^r$ and $pk = \left(g_1^z\right)^r$. The CA provides a certificate for each user by signing the user's public key together with its identity. The rest of the protocol works exactly as in the identity-based protocol instead that the public key of the communication partner is used to derive the session key. For example, Alice can compute her message a and the intermediate key k as $a = g_1^x sk_A$ for random $x \in \mathbb{Z}_p$ and $k = (e(b, g_2^z)/e(pk_B, g_2))^x$.

C mOT—the modified Okamoto—Tanaka protocol

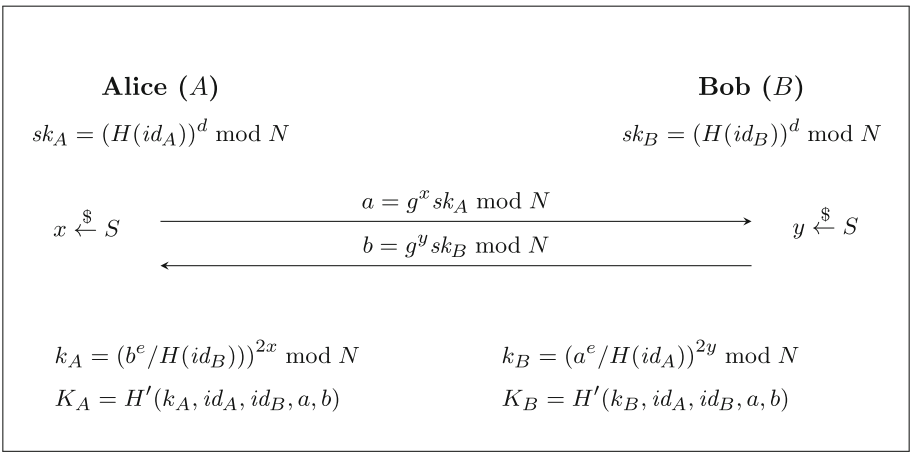

Fig. 4 Overview of mOT. The key generation center maintains public parameters *mpk* containing $N = p_1 p_2$ (for safe primes p_1, p_2), $g \in QR$, public exponent e with $\gcd(e, \phi(N)) = 1$, and descriptions of two hash functions $H : \{0,1\}^* \to QR$ and $H' : \{0,1\}^* \to \{0,1\}^*$. These parameters are available to all parties. The master secret *msk* consists of d with $ed = 1 \bmod \phi(N)$. The set of exponents can be set to $S = [1 \ldots (N-1)/4]$. It can be shown that this distribution is indistinguishable from $S = \left[1 \ldots \left\lfloor \sqrt{N}/2 \right\rfloor\right]$ under the factoring assumption [21]. In [20], the authors recommend a more aggressive choice of S with $S = \left[1 \ldots 2^{2\kappa}\right]$ and additionally assume that this distribution is indistinguishable from the previous ones. The master secret *msk* is used by the key generation center to derive the user secret keys as $sk_i = (H(id_i))^d$

D The uber-assumption

The non-interactive (k, l)-CBDHI and (k, l)-GCBDHI assumptions, can be viewed as special instantiations of Boyen's Uber-assumption (and its extensions) restricted to univariate polynomials. Following Boyen, our assumptions differ from the "classical" Uber-assumption in two ways. First, we consider computational assumptions (which are however implied by their decisional variants) and second we use rational exponents. However, in [8] Boyen also presents several extensions to the classical Uber-assumption that also cover these classes of assumptions. To apply Boyen's master theorem and show security in the generic bilinear group model we have to show independence of the polynomial $1/z$ (respectively $(z+w)/z^2$) over $\mathbb{Z}_p$ from $1, z, z^2, \ldots, z^k$ and $1, z, z^2, \ldots, z^l$. This means that there do not exist $(k+1)(l+1)$ constants $\{a_{i,j}\}$ for $i \in [0, k]$ and $j \in [0; l]$ such that for all $z \neq 0$ we always have

$$1/z = \sum_{i=0}^{k} \sum_{j=0}^{l} a_{i,j} z^{i+j}$$

or

$$(z+w)/z^2 = \sum_{i=0}^{k} \sum_{j=0}^{l} a_{i,j} z^{i+j}$$

over $\mathbb{Z}_p$. This is simple. These two equations are equivalent to

$$\sum_{i=0}^{k}\sum_{j=0}^{l} a_{i,j} z^{i+j+1} - 1 = 0$$

and

$$\sum_{i=0}^{k}\sum_{j=0}^{l} a_{i,j} z^{i+j+2} - z - w = 0.$$

Now for any choice of the $a_{i,j}$, the polynomials on the left-hand side have at least degree 1 (respectively 2). The maximal degree is $k+l+1 << p$ (respectively $k+l+2 << p$). This means they have at most $k+l+1$ (respectively $k+l+2$) roots. Thus the equations cannot be fulfilled for all $z \neq 0$. As a consequence we can apply the master theorem of Boyen to show that our assumptions are secure in the generic bilinear group model. Similarly, we can show that the CBDH assumption is secure in the generic bilinear group model. This time, we have to show that there do not exist constants a_0, a_1, a_2 with

$$xy = a_0 + a_1 x + a_2 y$$

for all $x, y \in \mathbb{Z}_p$. This is again very simple, as for any $x \neq a_2 \in \mathbb{Z}_p$ there is only a single $y \in \mathbb{Z}_p$ fulfilling the above equation, namely $y = (a_0 + a_1 x)/(x - a_2)$. Thus we can apply the master theorem and obtain security of this assumption in the generic bilinear group model. We stress that a successful adversary against the CBDH assumption can easily be used to break the DDH assumption in G_1: assume we are given the DDH challenge $g_1, g_1^a, g_1^b, h = g_1^{ab+c}$ and we have to decide whether $c = 0$. We can now use the CBDH attacker to compute $T = (g_1, g_2)^{ab}$. Next we compute $e(h, g_2)$ and check whether the result equals T. On success, we output that $c = 0$ otherwise $c = 1$.

E Krawczyk's impossibility result

In [24], Krawczyk presented a simple attack against the full PFS security of implicitly authenticated protocols. To illustrate it, assume a protocol in which Alice and Bob only exchange ephemeral Diffie-Hellman shares g^x (sent by Alice) and g^y (sent by Bob). Let us consider the situation where Alice acts as the initiator. To this end, she generates the ephemeral key g^x and sends it to Bob who responds with g^y. The adversary intercepts this value, and generates its own share by choosing random y and computing g^y. Then it sends g^y to Alice. Alice assumes this value was sent by Bob and generates the session key K from her secret key, x, and g^y. Now, assume that the adversary learns the secret key sk_B of Bob after Alice's session expired. Since the computation of the session key (from Bob's perspective) only depends on g^x and the knowledge of y and sk_B, the adversary is able to re-compute the session key K. Therefore it can always distinguish K from a random value. Observe that this impossibility result holds even in the case the two oracles have a common KGC that produces their secret keys or if they share a secret key. Essentially, the problem is that the adversary can always compute an ephemeral public key together with the corresponding ephemeral secret key that seems to come from Bob.

F Generalized impossibility result

Lemma 2 *Any two-message protocol which provides full PFS cannot allow the adversary to reveal ephemeral secret keys.*

Proof Without loss of generality assume Bob sends his message b first. Since the protocol provides (full) PFS by definition, any message b of Bob must contain an ephemeral public key epk. For contradiction assume the adversary can for one b reveal the corresponding ephemeral secret esk. The adversary can now easily break the security of the full PFS game by *replaying* b to the test-session (held by some party). It obtains back Bob's long-term secret sk_B. With esk, sk_B, and the message output by the test-session, the adversary can always derive the same session key as the test-session and thus win in the full PFS game. □

G FACTAS –factoring-based protocol with active security

The FACTAS protocol is defined in the group of signed quadratic residues [22]. It is a close variant of mOT with two major differences. The first one is that it is more efficient since there are no exponentiations by arbitrary exponents e'in the key derivation phase. Instead, FACTAS only considers squarings. The second difference is that the underlying security assumptions, though very close to those used in mOT, are more directly related to the factoring assumption instead of the (stronger) RSA assumption. In particular, KCI security and weak PFS are proven secure directly under the factoring assumption.

G.1 The group of signed quadratic residues

Let N be a safe integer, i.e. the product of two safe primes p_1 and p_2, each of bit-size κ. It is well-known that any safe prime larger than 5 always equals 3 modulo 4. Therefore, the product of two safe primes that are both larger than 5 is always a Blum integer. Let $\mathcal{J}_N$ be the multiplicative group of residues modulo N that have Jacobi symbol 1. Let $x \underline{\bmod} N \in (-N/2, N/2)$ be the representation of elements in $\mathbb{Z}_N^*$ as signed integers in the symmetric interval $(-N/2, N/2)$. We define $|x|$ to be the absolute value of $x \underline{\bmod} N$. In the following, we consider the group of signed quadratic residues $\mathcal{SQR}_N = \mathcal{J}_N \cap (0, N/2)$ with group operation $\circ$: $x \circ y = |xy|$ for $x, y \in \mathcal{SQR}_N$. Unless stated otherwise, all computations take place in this group. When it is clear from the context, we may write xy short for $x \circ y$ and x/y for $x \circ y^{-1} = x \circ y^{-1 \bmod |\mathcal{SQR}_N|} = \left|xy^{-1} \underline{\bmod} N\right|$ with $|\mathcal{SQR}_N| = |\mathbb{Z}_N^*|/4$. Similarly, we define $g^{\underline{a}} = g^{\underline{a} \bmod |\mathcal{SQR}_N|} = |g^a \underline{\bmod} N|$ for integer a. We stress that in $\mathcal{SQR}_N$ squaring is a permutation. We will also make use of the following fact.

Lemma 3 *Assume N is a safe integer. Assume g is a random generator of $\mathcal{SQR}_N$. Assume x is a uniformly random* odd *integer in* $[0, (N-1)/2]$. *Then $g^{\underline{x}}$ is statistically close to uniform in $\mathcal{SQR}_N$.*

Proof Let $N = p_1 p_2 = (2p_1' + 1)(2p_2' + 1) = 4p_1'p_2' + 2(p_1' + p_2') + 1$. First observe that $|\mathcal{SQR}_N| = \phi(N)/4 = (p-1)(q-1)/4 = p_1'p_2'$ is odd. At the same time drawing odd integers uniformly from $[0; (N-1)/2] = [0; 2p_1'p_2' + (p_1' + p_2')]$ is statistically close to drawing them from $[0; 2|\mathcal{SQR}_N|] = [0; 2p_1'p_2']$. Furthermore, since $p_1'p_2'$ is odd, the odd integers in $[p_1'p_2'; 2p_1'p_2']$ are equivalent to the even integers in $[0; p_1'p_2']$ modulo $|\mathcal{SQR}_N| = p_1'p_2'$. Also, given $|g^x|$, it is information-theoretically hidden whether x has been drawn as an even integer from $[0; p_1'p_2']$ or as an odd integer from $[p_1'p_2'; 2p_1'p_2']$. □

Also we will use the following well-known fact.

Lemma 4 *(Shamir's Trick) Assume we are given N, $g, h \in \mathbb{Z}_n^*$, and two integers a, b with $\gcd(a, b) = 1$ such that $g^a = h^b \bmod N$. Then we can easily compute $g^{1/b} \bmod N$ and $h^{1/a} \bmod N$.*

Definition 17 (FAC) Let N be a safe integer such that $N = pq$. The factoring assumption (FAC) states that no PPT $\mathcal{A}$ can output p and q when given N.

G.2 Security assumptions for the proof of full PFS security of FACTAS.

In our security proof for full PFS we will rely on the knowledge of exponent assumption for Diffie-Hellman in $\mathcal{SQR}_N$. This assumption was first introduced by Damgård in [13].

Definition 18 (KEA1) Let N be a safe integer and $\mathcal{SQR}_N$ the group of signed quadratic residues modulo N. Let $\mathcal{A}$ be a PPT that when given $X, Y \in \mathcal{SQR}_N$ outputs two group elements $X', Y' \in \mathcal{SQR}_N$. Then there exists another PPT algorithm $\mathcal{A}$' which given the same input and random coins as $\mathcal{A}$ outputs X', Y' and an integer s such that $X' = X^{\underline{s}}$ and $Y' = Y^{\underline{s}}$ if such an integer s exists.

Let us present the second security assumptions our security proof of full PFS security relies on. It is very similar to the Modified Knowledge of Exponent assumption for Discrete Logarithm introduced in [20].

Let N be a safe integer and $\mathcal{SQR}_N$ the group of signed quadratic residues modulo N. Consider the following security game between challenger $\mathcal{C}$ and PPT attacker $\mathcal{A}$.

1. The challenger $\mathcal{C}$ sends N, g and $U = g^{\underline{u}}$ to $\mathcal{A}$ where g and U are uniformly random in $\mathcal{SQR}_N$.
2. The attacker $\mathcal{A}$ sends $V \in \mathcal{SQR}_N$.
3. The challenger $\mathcal{C}$ outputs $U^{\underline{1/2}}$ and $z \in \mathbb{Z}_{|\mathcal{SQR}_N|}$ such that $g^{\underline{z}} = UV$.
4. $\mathcal{A}$ wins if it outputs an integer u'.

We say that $\mathcal{A}$ wins if $u' = u$.

Definition 19 (Modified Knowledge of Exponent Assumption for Discrete Log in $\mathcal{SQR}_N$ (MKEA-DLSQR)) For every PPT algorithm $\mathcal{A}$ in the above security game there exists another algorithm $\mathcal{A}$' which given the same inputs and random coins as $\mathcal{A}$' behaves exactly like $\mathcal{A}$ in the above security game but besides u' also outputs integers i, j with $V = g^{\underline{i}} U^{\underline{j}}$ whenever $\mathcal{A}$ wins.

In contrast to our assumption, the original Modified Knowledge of Exponent Assumption for Discrete Log introduced in [20] considers elements in $\mathcal{QR}$. Also $\mathcal{A}$ outputs $U^{1/e} \bmod N$ in the third step of the security game, where e is a public RSA exponent. However, the motivation behind our assumption and that of [20] are essentially equal.

H Security of FACTAS

The security proof follows the one of the mOT protocol.

Theorem 7 *In the random oracle model,* FACTAS *(Fig. 3) provides security against KCI and reflection attacks under the factoring assumption.*

Proof It is again easy to verify that two matching sessions compute the same key. Since they are matching they compute the same session identifier. Also they compute the same values k. Thus all inputs to H' are identical for each session and the session key is equal too.

Let us now show indistinguishability of real session keys from random keys. Assume we are given a factoring challenge consisting of the safe integer N'. The simulator sets $N = N'$, which implicitly sets *msk* to be the factorization of N. Additionally, the simulator chooses a random element h' from $[1, \ldots, (N-1)/2] \setminus \mathcal{SQR}_N$ and sets $h = h'^{\underline{2}}$ and $g = h^{\underline{2}}$. Observe that we have with overwhelming probability that h generates $\mathcal{SQR}_N$. As before, the simulator guesses the test-session among the set of all sessions with non-negligible success probability. Similarly, the simulator guesses, with non-negligible success probability, Bob to be the peer of the test session among the set of all parties. As in the previous proof, we ensure that the peer (Bob) of the test-session which is held by either Alice≠Bob (in case of KCI attacks) or Bob (when dealing with reflection attacks) always remains uncorrupted.

Setup and simulation of corrupt, register, reveal queries To be able to answer Corrupt queries for any party except Bob the simulator programs the outputs of the random oracle H for all inputs except for id_B as follows: given input id_i as input to a Register query it chooses a random value $r_i \in \mathcal{SQR}_N$ and outputs $H(id_i) := |r_i^2|$. In this way, the simulator can always compute a corresponding secret key as $sk_i = r_i$, simulate Register queries, and answer Corrupt queries. However, for id_B it chooses random odd $r_B \in [0; N/2]$ and sets $H(id_B) = h^{\underline{-r_B}}$. Observe that by Lemma 3, $H(id_B)$ is statistically close to uniform in $\mathcal{SQR}_N$.

In this way the simulator does not know the corresponding secret key of Bob which must be in $\mathcal{SQR}_N$. As in the previous proof, in almost all protocol runs the simulator makes the sessions (except for those whose holder is Bob) compute their messages and keys as specified in the protocol description. In this way it can also answer all Reveal queries (because the simulator knows the secret key of any party except for Bob).

To compute messages in sessions where Bob is session holder (we use b to denote the messages produced by these sessions), it does the following. The simulator chooses a random odd $b' \in [0; N/2]$ and computes $b = h^{\underline{b'}}$ (which again is statistically close to uniform in $\mathcal{SQR}_N$ by Lemma 3). It then holds that

$$b^{\underline{2}}/H(id_B) = h^{\underline{2b'+r_B}} = g^{\underline{b'}}h^{\underline{r_B}} = g^{\underline{b'+r_B/2}}.$$

Observe that now the secret exponent y in

$$b = h^{\underline{b'}} = g^{\underline{b'/2}} = g^{\underline{y}}(H(id_B))^{\underline{1/2}} = g^{\underline{y}}h^{\underline{-r_B/2}} = g^{\underline{y}}g^{\underline{-r_B/4}}$$

is not known to Bob (i.e. the simulator) as

$$y = b'/2 + r_B/4.$$

As a consequence, the simulator cannot compute k on behalf of Bob anymore when only given message a in case a is produced by the adversary in an active attack.

Simulating reveal queries for bob Let us show now how the simulator can successfully simulate sessions (and in particular Reveal queries) involving Bob (and the adversary). To this end, we first show that, although the simulator cannot compute $k^* = k$, it can nevertheless always compute $\overline{k} = k^{\underline{4}}$ even when the adversary $\mathcal{A}$ makes Bob engage in a communication with Bob himself. Recall that

$$k = \left(\frac{a^{\underline{2}}}{H(id_B)}\right)^{y}.$$

Now, independent of whether a has been computed by Bob (when considering reflection attacks), a session of any other party, or the adversary, the simulator can easily compute

$$k^{\underline{4}} = \left(\left(\frac{a^{\underline{2}}}{H(id_A)}\right)^{\underline{y}}\right)^{\underline{4}} = \left(\left(\frac{a^{\underline{2}}}{H(id_A)}\right)^{\underline{b'/2+r_B/4}}\right)^{\underline{4}} = \left(\frac{a^{\underline{2}}}{H(id_A)}\right)^{\underline{2b'+r_B}}.$$

Let us next describe the strategy of the simulator to program the second random oracle, H', and answer Reveal queries (including sessions whose holder or peer is id_B). Again we manage two sets R, S which are initially both empty. We first present our basic strategies. Then we show how we apply additional checks to maintain consistency. As in the previous proofs, our basic strategy is the following: whenever the attacker queries the random oracle with input $x_i = (k, id_A, id_B, a, b)$, we look up if there is some entry (x_i, y_i) already in R. In case it is not, we compute and output a new uniformly random string y_i and add (x_i, y_i) to R. If, on the other hand, (x_i, y_i) is already in R we simply output y_i.

To compute a session key (and answer a Reveal query) we proceed similarly. We look up if there is some entry (u_i, v_i) with $u_i = (id_A, id_B, a, b)$ already in S. In case it is not, we generate and output a new random string v_i and store (u_i, v_i) in S. If (u_i, v_i) is already in S we output v_i.

To make sure that the answers stored in S and R remain consistent we need to perform additional checks. We first recall that in some cases the simulator cannot compute k as pointed out before. Instead, the simulator computes $\overline{k} = k^{\underline{4}}$ for each session as shown above. (We remark that since it knows the corresponding secret keys, the simulator can actually compute k for the sessions of all parties except for Bob. However, our simulation strategy does not require that.) Now, whenever we receive a query $x_i = (k, id_A, id_B, a, b)$ we additionally check if $k \in \mathcal{SQR}_N$ and whether there is a corresponding element (u_j, v_j) in S with $u_j = (id_A, id_B, a, b)$ such that $k^{\underline{4}} = \overline{k}$ for the $\overline{k}$ value of that session. On success we output $y_i = v_j$ as stored in S. Otherwise we output a random y_i.

On the other hand, whenever we encounter a Reveal query $u_i = (id_A, id_B, a, b)$ for some session we can always compute $\overline{k}$ corresponding to u_i. Next we also check whether there is some entry (x_j, y_j) in R with $x_j = (k, id_A, id_B, a, b)$ and $k^{\underline{4}} = \overline{k}$ and $k \in \mathcal{SQR}_N$. On success, we output $v_i = y_j$ as stored in R. Otherwise we output a random v_i.

We remark that checking if $k \in \mathcal{SQR}_N$ is crucial: we show below that without this check, there exists an attacker that can tell the real security experiment and the simulation apart. This is one of the main reasons why we need to work in $\mathcal{SQR}_N$.

Extraction Now that we have showed how to simulate all attack queries, let us proceed to showing how the simulator extracts a solution to the factoring challenge. Either the test session is held by Alice≠Bob or Bob.

First we show how the simulator can extract a solution if the test-session is held by Alice. For this session we deviate from the general simulation strategy as described above. Instead of generating a honestly as $a = g^x H(id_A)^{\underline{1/2}}$ the simulator computes a as

$$a = h^{\underline{a'}} H(id_A)^{\underline{1/2}} = h^{\underline{a'+r_A}}$$

for some random odd $a' \in [0; N/2]$. Observe that now the discrete logarithm x in $a = g^{\underline{x}} H(id_A)^{\underline{1/2}} = g^{\underline{x}} h^{\underline{r_A}}$ is implicitly set to $x = a'/2$.

Assume the adversary has non-negligible success probability when querying the Test query to this session. In particular, it can decide whether the key provided by the Test query is the real session key or a random key from the same key space. We know that, since H' is modeled as a random oracle, the attacker must ask the real value k that is also computed by the test session to H' to have any probability significantly better than 1/2. With $y = b'/2 + r_B/4$

and $x = a'/2$, the simulator in this way obtains k such that

$$\begin{aligned} k &= \left(a^{\underline{2}}/H(id_A)\right)^{\underline{y}} \\ &= g^{\underline{2xy}} \\ &= g^{\underline{2(a'/2)(b'/2+r_B/4)}} \\ &= g^{\underline{a'(b'/2+r_B/4)}} \\ &= h^{\underline{a'(b'+r_B/2)}}. \end{aligned}$$

Also note that $k \in \mathcal{SQR}_N$. From this we can easily compute $h^{\underline{a'r_B/2}} = kh^{\underline{-a'b'}}$. This is equivalent to $h^{\underline{a'r_B}} = \left(kh^{\underline{-a'b'}}\right)^{\underline{2}}$. Now since $\gcd(2, a'r_B) = 1$, we can use Shamir's trick to compute $h^{\underline{1/2}}$. We now have, besides h' another, distinct root of h and $\gcd(h' - h^{\underline{1/2}}, N)$ gives a factor of N.

Let us now show how to extract a solution to the factoring challenge in case the test-session is held by Bob. In this case we have that $a = h^{\underline{a'}}$ and $b = h^{\underline{b'}}$ for some random odd a', b' such that $x = a'/2 + r_B/4$ and $y = b'/2 + r_B/4$. This time the simulator obtains the value k from the queries to the random oracle such that

$$\begin{aligned} k &= g^{\underline{2xy}} && (2) \\ &= g^{\underline{2(a'/2+r_B/4)(b'/2+r_B/4)}} && (3) \\ &= h^{\underline{(a'+r_B/2)(b'+r_B/2)}} && (4) \\ &= h^{\underline{a'b'+(a'+b')r_B/2+(r_B)^2/4}}. && (5) \end{aligned}$$

From this we can easily compute $h^{(r_B)^2/2} = \left(kh^{\underline{-a'b'-(a'+b')r_B/2}}\right)^{\underline{2}} \in \mathcal{SQR}_N$ which is equivalent to $h^{(r_B)^2} = \left(kh^{\underline{-a'b'-(a'+b')r_B/2}}\right)^{\underline{4}}$. Since $\gcd(4, (r_B)^2) = 1$, Shamir's trick again gives $h^{\underline{1/2}}$ and similarly to before, $\gcd(h' - h^{\underline{1/2}}, N)$ gives a factor of N. This concludes the proof. □

H.1 Weak PFS

We will now show that FACTAS provides weak PFS under the factoring assumption. Next we show that it also provides enhanced weak PFS under a variant of the Computational Diffie-Hellman assumption.

Lemma 5 FACTAS *provides weak PFS under the factoring assumption.*

Proof The setup is similar to the previous proof except that the simulator will know the secret key of Bob as well. The simulator uses the modulus N' from the factoring challenge as the modulus N of the scheme. It will program the random oracle such that it knows the secret keys of all users. To this end, it programs *H for all inputs* as follows: given input id_i as input to a Register query it chooses a random value $r_i \in \mathcal{SQR}_N$ and outputs $H(id_i) := |{r_i}^2|$. In this way, the simulator can always compute a corresponding secret key as $sk_i = r_i$, simulate Register queries, and answer Corrupt queries. Additionally, the simulator chooses a random element h' from $[1, \ldots, (N-1)/2] \setminus \mathcal{SQR}_N$ and sets $h = h'^{\underline{2}}$ and $g = h^{\underline{4}}$. Observe that we have with overwhelming probability that h generates $\mathcal{SQR}_N$. Now for all sessions, except for the test-session and its matching session (which must exist by the definition of weak PFS), the simulator computes the protocol messages and session keys as specified by the protocol

description. However for Alice's test-session and the corresponding, matching session of Bob it computes the protocol messages differently as

$$a = h^{\underline{a'}} sk_A$$
$$b = h^{\underline{b'}} sk_B$$

for random, odd integers a' and b' in $[0, (N-1)/2]$. This implicitly sets

$$x = a'/4$$
$$y = b'/4.$$

By Lemma 3 the resulting messages are statistically close to uniform elements in $\mathcal{SQR}_N$. The corresponding session key is thus derived from

$$k = g^{\underline{2xy}} = g^{\underline{a'b'/8}} = h^{\underline{a'b'/2}}.$$

Since the adversary can by assumption distinguish the session key from a random key, k must have been asked to the random oracle H' and thus be available to the simulator. Now we can easily find two distinct roots of $h^{\underline{a'b'}}$—$k \in \mathcal{SQR}_N$ and $h'^{a'b'} \notin \mathcal{SQR}_N$—and compute the factorisation of N'. □

H.2 Enhanced weak perfect forward secrecy

We can also show that FACTAS provides enhanced weak PFS albeit not under the factoring assumption. The problem is that in the security game the adversary is also given the master secret of the KGC—the factorisation of N and thus security cannot rely on the secrecy of this factorisation. We will now present a variant of the well-known Computational Diffie-Hellman assumption that allows us to nevertheless prove that FACTAS provides enhanced weak PFS. The main difference to the traditional Computational Diffie-Hellman assumption is that the challenge also consists of a safe prime p (such that $p = 2p' + 1$ for prime p') and all computations have to be performed in the subgroup of $\mathbb{Z}_p$ of order $p - 1/2$.

Computational Diffie-Hellman (CDHSP) Assumption in Groups of Safe Prime Order The CDHSP problem is, given safe prime $p = 2p' + 1$ an element g' of order p' and $g'^{x'}$, $g'^{y'}$ for random x', $y' \in [0; p' - 1]$ to compute $g'^{x'y'} \bmod p$.

Definition 20 (CDHSP Assumption) We say that attacker $\mathcal{A}$ breaks the CDHSP assumption if $\mathcal{A}$ succeeds in solving the CDHSP problem (where the probability is over the random coins of $\mathcal{A}$ and the random choices for p, g' and x', y'). We say that the CDHSP assumption holds if no PPT attacker $\mathcal{A}$ can break the CDHSP problem.

Lemma 6 FACTAS *provides enhanced weak PFS under the* CDHSP *assumption.*

Proof The idea is to make the simulator compute another safe prime q, set $N = pq$ and work in $\mathcal{SQR}_N$. Since the simulator knows the factorisation of N it can use the Chinese Remainder Theorem to compose and decompose elements in $\mathcal{SQR}_N$ into its components modulo p and modulo q. It chooses a random generator g of $\mathcal{SQR}_N$ such that the mod p-component is a known power of g'. Again the overall strategy of the simulator in the simulation phase is to set up everything honestly except for the ephemeral secrets of the test-session and its matching session. For these sessions it will embed $g'^{x'}$ and $g'^{y'}$ in the mod p-components of g^x and g^y. The value $k = g^{2xy}$ that the adversary queries to the random oracle H' must contain $g'^{2x'y'}$

in the mod p-component. The simulator can now use its knowledge of p to compute $g'^{x'y'}$ and break the CDHSP assumption. □

H.3 On the necessity of $\mathcal{SQR}_N$.

We would like to give a brief explanation on why it is not enough to work over the quadratic residues $\mathcal{QR}$ in $\mathbb{Z}_n^*$, i.e. why we really need to work in the efficiently recognizable group $\mathcal{SQR}_N$. To this end let us describe a hypothetical, successful, and unbounded attacker $\mathcal{A}$ which recognizes our simulation in case we work over $\mathcal{QR}$ and always aborts, thus making our extraction strategy fail. Attacker $\mathcal{A}$ behaves as follows:

- After receiving the public parameters, it factors N.
- Next $\mathcal{A}$ makes every party id_i engage in $t = t(\kappa)$ communications with $\mathcal{A}$, i.e. each party holds t sessions with $\mathcal{A}$ (i.e. some party controlled by $\mathcal{A}$). For each of these sessions $\mathcal{A}$ computes the corresponding intermediate value k. Among theses parties there must also be Bob for whom the simulator does not know the corresponding secret key. This means that the simulator is not able to compute the intermediate key k on its own. Instead, in the above security proof, the simulator computes k^4 and entirely relies on recognizing k when k is queried to the random oracle.
- Now, $\mathcal{A}$ does use its knowledge of the factorization of N to compute for every session another value $k' \in \mathbb{Z}_n^*$ such that $k' \neq k$ but $(k')^4 = k^4 \bmod N$.
- For each session, the adversary asks a Reveal query and receives output y.
- Now, $\mathcal{A}$ draws a random coin $c \in \{0, 1\}$ for each session. In case $c = 0$, $\mathcal{A}$ queries k and the corresponding identities and messages (as specified by the protocol) to the random oracle H'. However if $c = 1$, $\mathcal{A}$ queries k' to H'.
- The adversary aborts if i) at some point the random oracle outputs y (i.e. the response computed by the Reveal query) although $c = 1$ or ii) if the random oracle does not output y in case $c = 0$.
- The adversary makes two parties engage in a communication and correctly answers the Test query.

Basically, the attacker exploits, that the simulator which is not able to compute the intermediate values k for Bob cannot distinguish between the real intermediate value k and k'. However, in the real security game Bob is able to compute k and tell it apart from k'. Since the choices of the c values are hidden from the simulator, the probability that the simulator can correctly guess them all is 2^t and thus negligible. So in the real security game where the parties know their secret keys, the simulator never aborts. However, in the simulation it aborts with overwhelming probability.

A second reason why we use $\mathcal{SQR}_N$ instead of $\mathcal{QR}$ is that H must map identities to squares $x^2 \bmod N$. Since $\mathcal{QR}$ is not efficiently recognizable it is not clear how to accomplish that when mapping identities to $\mathcal{QR}$ (without revealing x in the process). However, as it is efficiently recognizable, mapping to $\mathcal{SQR}_N$ is simple. Intuitively, the hash function H can for example repeatedly generate "random-looking" strings r and check whether they belong to $\mathcal{SQR}_N$ when interpreted as a signed integers in $[-(N-1)/2, \ldots, (N-1)/2]$.

H.4 Efficiency of FACTAS

In our factoring-based protocol, message generation speed and message size are comparable to the mOT protocol. Computing the intermediate value k is more efficient since we only

need a single squaring in contrast to an exponentiation with the public RSA exponent e and a squaring. This makes the computation of the session key more efficient.

H.5 Full PFS security

Theorem 8 *Under the factoring assumption, the* KEA1 *assumption and the* MKEA-DLSQR *assumption the protocol depicted in Fig. 3 provides full PFS if we model H, H' as a random oracle.*

The proof of full PFS security very closely follows the proof of full PFS security in [20] and is therefore omitted. The main difference is that we now need to receive secret keys from the challenger of the MKEA-DLSQR assumption which are of the form $H(id_A)^{\frac{1}{2}}$ instead of $H(id_A)^{1/e}$ in [20]. This is exactly the main reason why we introduce our new variant of the modified knowledge of exponent assumption for discrete log.

Acknowledgements The author has been supported by the CONFIDENTIAL6G project that is co-funded by the European Union (grant agreement ID: 101096435).

Declarations

Competing interest The author does not have competing interests.

References

1. Abdalla, M., Bellare, M., Rogaway, P.: The oracle Diffie-Hellman assumptions and an analysis of DHIES. Topics in Cryptology-CT-RSA 2001: The Cryptographers' Track at RSA Conference 2001 San Francisco, CA, USA, April 8—12, 2001 Proceedings. Springer, Berlin (2001).
2. Barreto P.S.L.M., Naehrig M.: Pairing-friendly elliptic curves of prime order. In: Preneel B., Tavares S.E. (eds.) Selected Areas in Cryptography, pp. 319–331. Lecture Notes in Computer Science. Springer, New York (2005).
3. Bergsma, F., Jager, T., Schwenk, J.: One-round key exchange with strong security: An efficient and generic construction in the standard model. Public-Key Cryptography–PKC 2015: 18th IACR International Conference on Practice and Theory in Public-Key Cryptography, Gaithersburg, MD, USA, March 30–April 1, 2015, Proceedings 18. Springer, Berlin (2015). https://doi.org/10.1007/978-3-662-46447-2_21
4. Boneh D., Lynn B., Shacham H.: Short signatures from the Weil pairing. In: International conference on the theory and application of cryptology and information security. Berlin: Springer **17**(4), pp. 297–319 (2004).
5. Boneh, D., Gentry, C., Lynn, B., Shacham, H.: Aggregate and verifiably encrypted signatures from bilinear maps. In: Advances in Cryptology—EUROCRYPT 2003: International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4—8, 2003 Proceedings 22. Springer Berlin (2003).
6. Boneh D., Boyen X.: J. Cryptol. Efficient selective identity-based encryption without random oracles **24**(4), 659–693 (2011).
7. Boyd, C., Nieto, J.G.: On forward secrecy in one-round key exchange. In: Chen, L. (ed.) Cryptography and Coding-13th IMA International Conference, IMACC 2011, Oxford, UK, December 12–15, 2011.

Proceedings. Lecture Notes in Computer Science, vol. 7089, pp. 451–468. Springer, New York (2011). https://doi.org/10.1007/978-3-642-25516-8_27 . https://doi.org/10.1007/978-3-642-25516-8_27
8. Boyen, X.: The uber-assumption family (invited talk). In: 2nd International Conference on Pairing-based Cryptography (PAIRING 2008), volume 5209 of Lecture Notes in Computer Science, pp. 39–56(2008).
9. Canetti, R., Krawczyk, H.: Analysis of key-exchange protocols and their use for building secure channels. In: International conference on the theory and applications of cryptographic techniques. Berlin: pp. 453–474. (2001).
10. Cash D., Kiltz E., Shoup V.: The twin Diffie-Hellman problem and applications. J. Cryptol. **8**, 470–504 (2008).
11. Chen Y., Huang Q., Zhang Z.: Sakai–Ohgishi–Kasahara identity-based non-interactive key exchange revisited and more. Int. J. Inform. Secur. **15**, 15–33 (2014). https://doi.org/10.1007/978-3-319-08344-5_18.
12. Cremers, C.J.F., Feltz, M.: Beyond eCK: Perfect forward secrecy under actor compromise and ephemeral-key reveal. Computer Security–ESORICS 2012: 17th European Symposium on Research in Computer Security, Pisa, Italy, September 10–12, 2012. Proceedings 17. Springer Berlin (2012).
13. Damgård, I.: Towards practical public key systems secure against chosen ciphertext attacks. Advances in Cryptology—CRYPTO'91: Proceedings 11. Springer, Berlin (1992).
14. Di Raimondo, M., Gennaro, R., Krawczyk, H.: Deniable authentication and key exchange, In: Proceedings of the 13th ACM conference on Computer and communications security. pp. 400–409 (2006).
15. Diffie W., Hellman M.E.: New directions in cryptography. IEEE Trans. Inform. Theory IT **22**(6), 644–654 (1976).
16. Feltz, M., Cremers, C.: On the limits of authenticated key exchange security with an application to bad randomness. IACR Cryptology ePrint Archive, 369 (2014).
17. Fiore, D., Gennaro, R.: Making the Diffie–Hellman protocol identity-based. Topics in Cryptology-CT-RSA 2010: The Cryptographers' Track at the RSA Conference 2010, San Francisco, CA, USA, March 1–5, 2010. Proceedings (2010).
18. Fischlin, M., Fleischhacker, N.: Limitations of the meta-reduction technique: The case of schnorr signatures. Annual International Conference on the Theory and Applications of Cryptographic Techniques. Berlin, Heidelberg, pp. 444–460. (2013). https://doi.org/10.1007/978-3-642-38348-9_27
19. Galbraith S.D., Paterson K.G., Smart N.P.: Pairings for cryptographers. Discret, Appl. Math. **156**(16), 3113–3121 (2008).
20. Gennaro, R., Krawczyk, H., Rabin, T.: Okamoto–Tanaka revisited: Fully authenticated Diffie–Hellman with minimal overhead. Applied Cryptography and Network Security: 8th International Conference, ACNS 2010, Beijing, China, June 22–25. Proceedings 8. Springer Berlin (2010).
21. Goldreich O., Rosen V.: On the security of modular exponentiation with application to the construction of pseudorandom generators. Cryptology **16**(2), 71–93 (2003).
22. Hofheinz, D., Kiltz, E.: The group of signed quadratic residues and applications. Annual International Cryptology Conference. Berlin, Heidelberg, pp. 637–653. (2009).
23. Krawczyk, H.: SKEME: a versatile secure key exchange mechanism for internet. In: 1996 Symposium on Network and Distributed System Security, (S)NDSS '96, San Diego, CA, February 22–23, 1996, pp. 114–127 (1996).https://doi.org/10.1109/NDSS.1996.492418. http://doi.ieeecomputersociety.org/10.1109/NDSS.1996.492418
24. Krawczyk H.: HMQV: A High-performance Secure Diffie–Hellman Protocol, pp. 546–566. Springer, Berlin (2005).
25. Law L., Menezes A., Qu M., Solinas J.A., Vanstone S.A.: An efficient protocol for authenticated key agreement. Des. Codes Cryptogr. **28**(2), 119–134 (2003).
26. Okamoto E., Tanaka K.: Key distribution system based on identification information. IEEE J. Select. Areas Commun. **7**(4), 481–485 (1989). https://doi.org/10.1109/49.17711.
27. Ratnam K., Gurusamy M., Zhou L.: Differentiated survivability with improved fairness in ip/mpls-over-wdm optical networks. Comput. Netw. **53**(5), 634–649 (2009). https://doi.org/10.1016/j.comnet.2008.11.001.
28. Rifá-Pous H., Herrera-Joancomartí J.: Computational and energy costs of cryptographic algorithms on handheld devices. Future Internet **3**(1), 31–48 (2011). https://doi.org/10.3390/fi3010031.
29. Shoup, V.: On Formal models for secure key exchange. Cryptology ePrint Archive, Report 1999/012. http://eprint.iacr.org/ (1999).
30. Sivakumar N.R., Nagarajan S.M., Devarajan G.G., Pullagura L., Mahapatra R.P.: Enhancing network lifespan in wireless sensor networks using deep learning based graph neural network. Phys. Commun. **59**, 102076 (2023). https://doi.org/10.1016/j.phycom.2023.102076.

31. Song K., Wang Q., Peng L., Li C., Wu X.: Secrecy energy efficiency optimization for df relaying iot systems with passive eavesdropping terminal. Phys. Commun. **44**, 101254 (2021). https://doi.org/10.1016/j.phycom.2020.101254.