TOPAS

2-Pass Key Exchange with Full Perfect Forward Secrecy and Optimal Communication Complexity

Sven Schäge
Ruhr-Universität Bochum
Germany
sven.schaege@rub.de

ABSTRACT

We present TOPAS (Transmission Optimal Protocol with Active Security), the first key agreement protocol with optimal communication complexity that provides security against *fully active* adversaries. This solves a longstanding open problem. The size of the protocol messages ($\approx$ 160 bits for 80-bit security) 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 – under generalizations of the Computational Bilinear Diffie-Hellman Inversion assumption. 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). 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.

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 [14] 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 [21] and its successor HMQV [20]. 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 [20]. Unfortunately, in 2005 Krawczyk presented an attack that shows that *implicitly-authenticated protocols inherently cannot provide forward secrecy against active attackers* [20] (see Appendix D 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 B for a brief example). 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 [18], the support of PFS is an important advantage over 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. We note that although RSA-based key transport is still the most common TLS variant in use, Google has recently announced that it will push the use of TLS with ephemeral Diffie-Hellman key exchange exactly because of its guarantee of perfect forward secrecy [2]. Moreover, it is clear that the next version of TLS will not support RSA-based key establishment partly because of its lack of forward secrecy [24]. 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) [18]. Basically, mOT is an enhanced variant of the classical Okamoto-Tanaka protocol [22] 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 assump-

CCS'15, October 12–16, 2015, Denver, Colorado, USA.

ACM 978-1-4503-3832-5/15/10 ...$15.00.
DOI: http://dx.doi.org/10.1145/2810103.2813683.

[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.

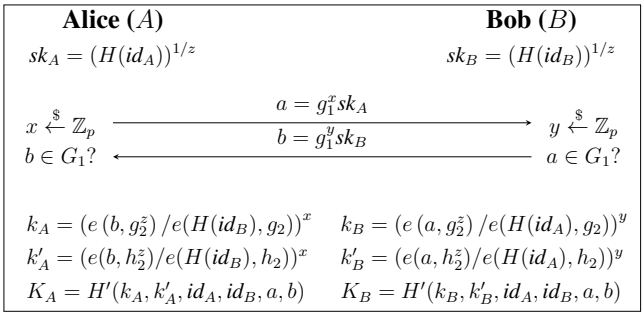

Figure 1: Overview of TOPAS. The key generation center maintains public parameters *mpk* containing $g_1, g_2, h_2, g_2^z, h_2^z$, prime p, a description of the pairing e, and descriptions of two hash functions $H : \{0,1\}^* \to G_1$ and $H' : \{0,1\}^* \to \{0,1\}^*$. These parameters are available to all parties. 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 divisions in G_T per key exchange. Messages can be sent in any order. Without loss of generality we assume that lexically $id_A \leq id_B$.

tion. Unfortunately, group elements consists 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 a longstanding open problem to design a protocol with full security (including full PFS) against active attackers *and* optimal communication complexity, i.e. where each message only consists of about 160 bits.[2] This is of course optimal, since the birthday bound requires messages to be at least 160 bits for 80 bit security.

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 (Figure 1). To achieve this, the design of TOPAS relies on new ideas and techniques that are different from all existing two message protocols that we are aware of. 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 C, 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-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. Remarkably, for 80 bit security, each message consists of only about 160 bits, 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 and thus optimal as well.

We also present, TOPAS+ (Figure 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 the session key. When precomputing 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 mentioned before, the identity-based properties of our protocols avoid that additional information like certificates have to be exchanged between unknown communication partners, in contrast to PKI-based protocols like for example HMQV. This guarantees that in TOPAS and TOPAS+ the size of each messages does indeed never exceed 160 bits. Also the time for key derivation is not slowed down by the additional verification of the received certificate. 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 [13]. 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.

[2]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.

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 [16] 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, 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. Due to space limitations we postpone the formal statement and proof of this result to the full version of this paper.

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.

Protocol	id-based (no cert.)	message bit-size	pairings in key derivat. on./off.	security assumptions		
				key indist. KCI, reflection	full PFS	ephemeral secret key reveal
HMQV [20]	no	160 (+ cert.)[3]	–	Gap DH (interactive)	–	extract.
mOT [18]	yes	1024	–	RSA	extract.	–
TOPAS	yes	160	4/2	generaliz. of CBDHI (non-inter.)	extract.	–
TOPAS+	yes	160	2/1	generaliz. of CBDHI, SDH oracle (interactive)	extract.	–

Table 1: Comparison of 2-pass key exchange protocols for 80-bit security.

APPLICATION SCENARIOS. Our protocols are interesting for communication 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. 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 [18]. 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 [20]. 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 [18]. 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. In Appendix E we show that any protocol which allows the adversary to reveal ephemeral keys, cannot provide full PFS.

[3]We note that whenever the communication partners exchange certificates in HMQV to authenticate their public keys this not only affects the message size but also the key derivation. In particular, the time for key derivation is increased by the time for the additional certificate verification. We stress that (long-lived) certificates only have to be transferred and checked once per communication partner.

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 [23].[4] This approach has been generalized in [12, 4]. 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 [19]). 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 [13]. 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 E. So, as the protocols in [12, 4] allow the adversary to reveal ephemeral keys, they cannot be shown to provide full PFS in the strong sense of [18].

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 [15]. 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

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

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 need to exchange certificates before communicating for the first time. In Appendix A we briefly sketch this transformation. We leave a formal security analysis of this protocol variant for future work.

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 $\log_2(p)$ be 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) [17]. 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 [7, 3].

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 C 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. 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 and $g_1^z, g_1^{z^2}, \ldots, g_1^{z^k}$, $g_2^z, g_2^{z^2}, \ldots, g_2^{z^l}$ 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 [5] where k is fixed to $k = 2$.

DEFINITION 1. *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-HELLMAN 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. *Adversary $\mathcal{A}$ 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 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_1. Assume $G = (p, g_1, g_2, e)$ 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. *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. *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 $\mathcal{A}$ 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 [6] 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 $g_2^z$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 the group elements $A^{1/z} \in G_1$ 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. *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 $\mathcal{A}$ 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 6. *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 $H_t(m_1) = H_t(m_2)$.*

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 [18].

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 and a *session key* K or *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.

- 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.
- 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 [18] 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 original model in [20] 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 [18] 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 E 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.

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.

SECURITY DEFINITIONS. Let **SG** 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, 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 7. *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 $1/2$.*

[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.

DEFINITION 8 (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 9 (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 $id_A = id_B$.*

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 10 (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 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. 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 11 (KCI SECURITY). *We say that Π is secure against KCI attacks if in* **SG** *attacker $\mathcal{A}$ is additionally given the secret key of the holder of the test session at the beginning of the security experiment.*

Obviously, KCI security implies security under Definition 7 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 GKR, 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 12 (SECURITY AGAINST REFLECT. 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 Figure 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 D), 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 strip off the signature sk_A (and thus any identity-specific information) from the message $a = g_1^x sk_A$. However, the result lies in the target group and has an additional exponent z:

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

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 this 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 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 7. (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

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

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

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

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

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

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 Equation 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 check 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 $x = a'/z$.

Suppose that 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 if 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

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

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 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 $k^{1/z} = e(g_1, g_2)^{xy}$. □

3.2 Proof of Full PFS Security

THEOREM 3. *The* TOPAS *protocol 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 stepwisely 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

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

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.

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 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 using the values provided to $\mathcal{A}_2$ by the PFS challenger and $\mathcal{A}_2$'s output values. We then argue that $\mathcal{A}_3$ always wins given that $\mathcal{A}_2$ wins against the PFS challenger. Let, $G, g_2^z, h_2, h_2^z, a = g_1^x H(id_A)^{1/z}, sk_A, H(id_B)$ be the values provided by the full PFS challenger to $\mathcal{A}_2$. The input to $\mathcal{A}_3$ is $G, g_2^z, B' = H(id_B)$. When $\mathcal{A}_2$ outputs $b, k^{1/x}$, $\mathcal{A}_3$ will output $Y' = k^{1/x}$ 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. Let us now show another adversary $\mathcal{A}_5$ that controls $\mathcal{A}_4$ and $\mathcal{A}_2$ to win against a PFS challenger while outputting additional values besides what is required by definition. By construction we have that the Modified Knowledge of Co-CDH attacker $\mathcal{A}_4$ uses the PFS attacker $\mathcal{A}_2$ as a (black-box) subroutine. In the following $\mathcal{A}_5$ will modify the communication between $\mathcal{A}_4$ and $\mathcal{A}_2$ 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 $H(id_B)$ to $\mathcal{A}_4$. 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 other attacker, $\mathcal{A}_4$, outputs $k^{1/x}$ together with i, T to $\mathcal{A}_5$. However, $\mathcal{A}_5$ 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 $g_2^* = g_2^{yz}$. Both 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

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

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 (Figure 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.

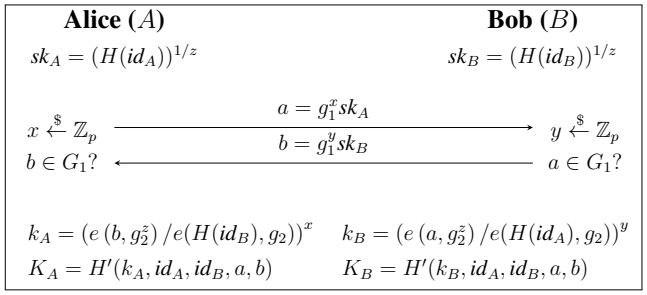

Figure 2: Overview of TOPAS+. The KGC maintains public parameters *mpk* containing g_1, g_2, g_2^z, p, a description of the pairing e, and descriptions of two hash functions $H : \{0,1\}^* \to G_1$ and $H' : \{0,1\}^* \to \{0,1\}^*$. These parameters are available to all parties. The master secret *msk*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

DEFINITION 13. *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.*

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.

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

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 [13].

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 *and* TOPAS+ *meet the strong notion of deniability of [13].*

PROOF. Observe that $a = g^x sk_A$ is uniformly distributed since x is uniform. Therefore the simulator can simulate Alice's message a by just choosing a random group element in G_1. 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. ACKNOWLEDGEMENTS

I am grateful to Yong Li, who generously allowed me to use his observations in the impossibility result of Appendix E. I would also like to thank the anonymous reviewers for their helpful comments. This work was supported by the DFG-Research Training Group UbiCrypt (GRK 1817/1).

7. REFERENCES

[1] Michel Abdalla, Mihir Bellare, and Phillip Rogaway. The oracle Diffie-Hellman assumptions and an analysis of DHIES. In David Naccache, editor, *CT-RSA 2001*, volume 2020 of *LNCS*, pages 143–158. Springer, April 2001.

[2] Google Security Team Adam Langley. Protecting data for the long term with forward secrecy. `http://googleonlinesecurity.blogspot.co.uk/2011/11/protecting-data-for-long-term-with.html`.

[3] Paulo S. L. M. Barreto and Michael Naehrig. Pairing-friendly elliptic curves of prime order. In Bart Preneel and Stafford E. Tavares, editors, *Selected Areas in Cryptography*, volume 3897 of *Lecture Notes in Computer Science*, pages 319–331. Springer, 2005.

[4] Florian Bergsma, Tibor Jager, and Jörg Schwenk. One-round key exchange with strong security: An efficient and generic construction in the standard model. In *PKC 2015*, LNCS, pages 477–494. Springer, 2015.

[5] Dan Boneh and Xavier Boyen. Efficient selective identity-based encryption without random oracles. *Journal of Cryptology*, 24(4):659–693, October 2011.

[6] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 416–432. Springer, May 2003.

[7] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. *Journal of Cryptology*, 17(4):297–319, September 2004.

[8] Xavier Boyen. The uber-assumption family (invited talk). In Steven D. Galbraith and Kenneth G. Paterson, editors, *PAIRING 2008*, volume 5209 of *LNCS*, pages 39–56. Springer, September 2008.

[9] Ran Canetti and Hugo Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 453–474. Springer, May 2001.

[10] David Cash, Eike Kiltz, and Victor Shoup. The twin Diffie-Hellman problem and applications. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 127–145. Springer, April 2008.

[11] Yu Chen, Qiong Huang, and Zongyang Zhang. Sakai-Ohgishi-Kasahara identity-based non-interactive key exchange revisited and more. In Willy Susilo and Yi Mu, editors, *ACISP 14*, volume 8544 of *LNCS*, pages 274–289. Springer, July 2014.

[12] Cas J. F. Cremers and Michele Feltz. Beyond eCK: Perfect forward secrecy under actor compromise and ephemeral-key reveal. In Sara Foresti, Moti Yung, and Fabio Martinelli, editors, *ESORICS 2012*, volume 7459 of *LNCS*, pages 734–751. Springer, September 2012.

[13] Mario Di Raimondo, Rosario Gennaro, and Hugo Krawczyk. Deniable authentication and key exchange. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 06*, pages 400–409. ACM Press, October / November 2006.

[14] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.

[15] Dario Fiore and Rosario Gennaro. Making the Diffie-Hellman protocol identity-based. In Josef Pieprzyk, editor, *CT-RSA 2010*, volume 5985 of *LNCS*, pages 165–178. Springer, March 2010.

[16] Marc Fischlin and Nils Fleischhacker. Limitations of the meta-reduction technique: The case of schnorr signatures. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 444–460. Springer, May 2013.

[17] Steven D. Galbraith, Kenneth G. Paterson, and Nigel P. Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, 156(16):3113–3121, 2008.

[18] Rosario Gennaro, Hugo Krawczyk, and Tal Rabin. Okamoto-Tanaka revisited: Fully authenticated Diffie-Hellman with minimal overhead. In Jianying Zhou and Moti Yung, editors, *ACNS 10*, volume 6123 of *LNCS*, pages 309–328. Springer, June 2010.

[19] Hugo Krawczyk. SKEME: a versatile secure key exchange mechanism for internet. In James T. Ellis, B. Clifford Neuman, and David M. Balenson, editors, *1996 Symposium on Network and Distributed System Security, (S)NDSS '96, San Diego, CA, February 22-23, 1996*, pages 114–127. IEEE Computer Society, 1996.

[20] Hugo Krawczyk. HMQV: A high-performance secure Diffie-Hellman protocol. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 546–566. Springer, August 2005.

[21] Laurie Law, Alfred Menezes, Minghua Qu, Jerome A. Solinas, and Scott A. Vanstone. An efficient protocol for authenticated key agreement. *Des. Codes Cryptography*, 28(2):119–134, 2003.

[22] Eiji Okamoto and Kazue Tanaka. Key distribution system based on identification information. *IEEE Journal on Selected Areas in Communications*, 7(4):481–485, 1989.

[23] Victor Shoup. On formal models for secure key exchange. Cryptology ePrint Archive, Report 1999/012, 1999. http://eprint.iacr.org/.
[24] Transport Layer Security working group of the IETF. Confirming consensus on removing RSA key transport from TLS 1.3. http://www.ietf.org/mail-archive/web/tls/current/msg12362.html.

APPENDIX

A. 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 $pk = (g_1^z)^r$. 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 $k = (e(b, g_2^z)/e(pk_B, g_2))^x$.

B. ON THE IMPORTANCE OF FULL PFS

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. 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 does so.

C. 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

$$1/z = \sum_{i=0}^{k}\sum_{j=0}^{l} a_{i,j} z^{i+j} \text{ or } (z+w)/z^2 = \sum_{i=0}^{k}\sum_{j=0}^{l} a_{i,j} z^{i+j}$$

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

$$\sum_{i=0}^{k}\sum_{j=0}^{l} a_{i,j} z^{i+j+1} - 1 = 0 \text{ or } \sum_{i=0}^{k}\sum_{j=0}^{l} a_{i,j} z^{i+j+2} - z - w = 0.$$

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 $c = 1$.

D. EXISTING IMPOSSIBILITY RESULT

In [20], 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.

E. NEW IMPOSSIBILITY RESULT

LEMMA 1. *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. □